child 2

EXTENDED BASIC

general processor

C H I L D E X T E N D E D B A S I C

Le informazioni contenute nel presente manuale sono state accuratamente verificate e si può ritenere che siano esatte. Non è assunta comunque nessuna responsabilità per eventuali inesattezze.

La General Processor si riserva, nell'interesse del continuo miglioramento della sua linea di prodotti, di effettuare qualsiasi modifica senza alcun preavviso.

Child è un marchio depositato della General Processor Firenze

general processor

SISTEMI DI ELABORAZIONE - MICROPROCESSORI
VIA MONTEBELLO, 3 - 3a rosso
TEL. 055 / 219.143 – 50123 FIRENZE

Testo originale di Stefano Giusti. - Copyright 1979 by General Processor - Firenze. Tutti i diritti sono riservati.
Finito di stampare nel Settembre 79 presso il Centro 2P - Firenze.

Copyright prima edizione , settembre 79 by General Processor - Firenze.

INTRODUZIONE E GENERALITA'

1-1 Introduzione al manuale

a. Convenzioni. Nel discutere le caratteristiche del Child Extended BASIC adotteremo per semplicità certe convenzioni, che riportiamo di seguito.

1. Le parole stampate in lettere maiuscole devono essere scritte esattamente come mostrato. Si tratta per lo più di nomi di istruzioni e comandi.

2. I termini racchiusi fra apici (') devono essere forniti assieme alla parola chiave come è spiegato nel testo. Quelli racchiusi fra i segni di grado (°) sono opzionali. Infine i termini seguiti da puntini (...) possono essere ripetuti più volte od eliminati a seconda dei casi.

3. Shift/ (o Control/) seguito da una lettera significa che il carattere si ottiene tenendo premuto il tasto Shift (o Control) e battendo la lettera indicata.

4. Tutta la punteggiatura indicata deve essere rispettata.

b. Definizioni. I seguenti termini diverranno importanti nel seguito:

Caratteri alfanumerici: tutte le lettere e i numeri presi insieme sono detti caratteri alfanumerici.

Ritorno carrello: si riferisce sia al tasto del terminale che provoca il posizionamento del carrello, della testina stampante o del cursore all'inizio della riga successiva, sia al comando, inviato appunto dal tasto ritorno carrello, che termina una linea BASIC.

Livello comandi: il BASIC si trova al livello comandi, è cioè pronto ad accettare comandi, dopo che ha stampato *.

Comandi e Frasi: le istruzioni BASIC sono divise strettamente in due classi, Comandi e Frasi. I comandi sono istruzioni usate normalmente solo in modo diretto (v. "Modi di funzionamento", par. 1-2). Alcuni comandi, come CONT, possono essere usati solo in modo diretto, non avendo significato come frasi

di programma. Altri, come DELETE, non sono in genere usati come frasi di programma, perchè provocano un ritorno al livello comandi. Ma la maggior parte dei comandi trova uso occasionale come frasi di programma.

Le frasi sono istruzioni normalmente usate solo in modo indiretto. Alcune, come DEF, possono essere usate solo in tale modo.

Edit: sia al processo di eliminare, aggiungere e sostituire frasi di programma, che a quello di preparare dati per l'uscita in accordo ad un formato prefissato ci riferiremo con il termine di editing. Il particolare significato in uso sarà chiaro dal contesto.

Espressione intera: un'espressione il cui valore è troncato ad un intero. Gli argomenti dell'espressione non devono necessariamente essere interi.

Parole riservate: alcune parole sono riservate dal BASIC come parole chiave di frasi e comandi. Queste parole sono dette riservate e non possono es-

sere usate nè come nomi di variabili e funzioni, nè come parti di nomi di variabili e funzioni.

Caratteri speciali: alcuni caratteri appaiono diversi su differenti terminali. Di questi i più importanti sono:

^ appare su certi terminali come

non appare su certi terminali dove stam- un blank

_ (sottolineatura) a volte appare come

Stringa letterale: una stringa di caratteri racchiusi fra virgolette ("), che deve essere letta od emessa esattamente come appare. Le virgolette non fanno parte della stringa, nè d'altra parte una stringa può contenerle. Ad esempio la stringa ""PIPPO"" non è consentita.

1-2 Modi di funzionamento

Il Child BASIC provvede alle operazioni dell'elaboratore in due modi diversi. Nel modo diretto le frasi o i comandi sono eseguiti non appena im-

messi nel computer. I risultati delle operazioni aritmetiche e logiche sono visualizzati e memorizzati per un eventuale uso successivo, ma le istruzioni sono perse subito dopo la loro esecuzione. Questo modo è utile per operazioni di debug e per usare l'elaboratore come "calcolatrice" per calcoli veloci che no giustificano la stesura di programmi completi.

nel modo indiretto l'elaboratore esegue istruzioni prelevandole successivamente da un programma residente in memoria. Le linee di programma sono memorizzate se precedute da un numero di linea. L'esecuzione del programma è iniziata dal comando RUN.

1-3 Formati

a. Linee. La linea è l'elemento fondamentale di un programma BASIC. Il suo formato è il seguente:

nnnnn 'frase BASIC'°:'frase BASIC'...°

Ogni linea inizia con un numero che corrisponde al-

l'indirizzo in memoria della linea. Il numero di linea è necessario anche per i salti di programma e per le operazioni di editing. Anche se i numeri di linea possono variare fra Ø e 65529, è buona norma usare un incremento di 5 o di 1Ø fra linee successive al fine di consentire eventuali inserzioni di nuove linee.

1. Nel Child BASIC i numeri di linea possono essere generati automaticamente tramite i comandi AUTO e RENUM. Il comando AUTO provvede all'inserzione automatica, in fase di stesura di un programma, dei numeri di linea. Il suo formato è:

AUTO °'linea iniziale'°°,°'incremento'°°

Esempio:

```
AUTO 1ØØ,1Ø
1ØØ INPUT X,Y
11Ø PRINT SQR(X^2+Y^2)
12Ø ^C
*
```

Il comando AUTO numera ogni linea in ingresso, finchè non è battuto Control/C. Se la linea iniziale è

omessa, è assunta uguale a 10, così come è assunto uguale a 10 l'incremento se omesso. Se infine il numero indicante la linea iniziale è seguito dalla virgola ma non dall'incremento, questo viene assunto pari all'ultimo incremento usato in una frase AUTO.

Se il comando AUTO genera un numero di linea già esistente nel programma correntemente in memoria, viene stampato il numero di linea seguito da un asterisco, per segnalare all'operatore che un suo ingresso da tastiera sostituirà la linea esistente.

2. Il comando RENUM permette di distanziare fra di loro i numeri di linea di un programma, in modo che una o più linee possano essere inserite fra quelle già esistenti. Il formato del comando é il seguente:

RENUM°'NN'°,'MM'°,'II'°°°

dove NN è il numero che viene attribuito alla prima linea nella nuova numerazione. Nel caso che NN sia omesso, viene assunto uguale a 10. Le linee minori di MM non vengono rinumerate, e, se MM è omesso, viene rinumerato l'intero programma. Infine II

è l'incremento che viene dato fra linea e linea nella nuova sequenza. Al solito è assunto pari a 10, se non è specificato.

Per maggiore chiarezza riportiamo alcuni esempi:

RENUM Rinumera l'intero programma, partendo dalla linea 10, con un incremento di 10 fra linea e linea.

RENUM 100,100 Rinumera l'intero programma a partire dalla linea 100, con un incremento di 100.

RENUM 6000,5000,1000 Rinumera le linee da 5000 in su, in modo che partano da 6000 con un incremento di 1000.

N.B. Il comando RENUM non puo essere usato per alterare l'ordine delle linee di un programma (ad es. RENUM 15,30 quando il programma ha tre linee numerate 10,20 e 30), nè per generare numeri di linea superiori a 65529. In tal caso si ha la segnalazione di errore ILLEGAL FUNCTION CALL.

Tutti i numeri di linea che appaiono dopo le

frasi GOTO, GOSUB, THEN, ON...GOTO, ON...GOSUB ed ERL'operatore di relazione' sono appropriatamente cambiati dal comando RENUM per riferirsi ai nuovi numeri di linea. Se dopo una delle frasi citate appare un numero di linea che non esiste nel programma, viene stampato il messaggio "UNDEFINED LINE XXXXX IN YYYYY". Il riferimento alla linea XXXXX non viene alterato dal comando RENUM, mentre il numero di linea YYYYY può essere cambiato.

3. Dovunque occorre un riferimento ad un numero di linea, il numero di linea corrente può essere designato semplicemente da un punto(.). Ciò risulta particolarmente utile nell'uso del comando EDIT (v. dopo).

4. Di seguito al numero di linea vengono scritte una o più frasi BASIC. In ogni frase la prima parola identifica l'operazione da svolgere. La lista degli argomenti che segue questa parola svolge diversi compiti. Essa può contenere i dati sui quali si devono svolgere le operazioni richieste, o a tali dati si può riferire simbolicamente. In alcune istruzioni molto importanti le operazio-

ni da eseguire dipendono da condizioni od opzioni specificate nella lista.

Su di una stessa linea possono essere scritte più frasi, separate fra di loro dai due punti. Il numero di frasi, che possono essere concatenate su di una stessa linea con questo metodo, è limitato solo dal numero complessivo dei caratteri, che deve risultare inferiore a 255. Le linee possono essere spezzate tramite il tasto LINE FEED (interlinea), come è mostrato nel seguente esempio:

```
100 IF X=Y+23'interlinea'
    THEN 5'interlinea'
    ELSE PRINT X'ritorno carrello'
```

Tutto questo, sebbene visualizzato su tre righe diverse, viene interpretato da BASIC come un'unica linea di programma.

b. Commenti (REMarks). In molti casi un programma può essere compreso più facilmente, se insieme alle frasi di programma vere e proprie contiene un certo numero di commenti e spiegazioni.

Nel BASIC la frase REM permette di inserire tali commenti senza naturalmente influenzare la normale esecuzione del programma. Il formato della frase è il seguente:

```
REM'commenti'
```

Si noti che, anche se il BASIC non esegue la frase REM, tuttavia le istruzioni di salto possono puntare ad essa. La REM è terminata dal ritorno carrello o dalla fine della linea, ma non dai due punti, come mostra il seguente esempio:

```
100 REM ESEGUI QUESTO CICLO:FOR I=1 TO 20
120 FOR I=1 TO 20:REM ESEGUI QUESTO CICLO
```

Nel primo caso la frase FOR non viene eseguita, ma è considerata parte del commento; nel secondo sì.

I commenti possono anche essere inseriti alla fine di una linea di programma, separati dalle frasi precedenti da un apice ('). Ogni carattere dopo l'apice sarà ignorato in fase di esecuzione.

c. Errori. Quando individua un errore tale da causare un'interruzione del programma, l'inter-

prete BASIC stampa un messaggio di errore. I formati di questi messaggi sono:

Frasi dirette	?XX ERROR
Frasi indirette	?XX ERROR IN nnnnn

dove XX è il codice o messaggio di errore ed nnnnn è il numero della linea in cui si trova l'errore. Ogni istruzione ha i suoi possibili errori specifici, oltre agli errori generali di sintassi. I vari errori saranno discussi nella descrizione delle singole istruzioni.

1-4 Primi elementi di Editing

Le possibilità di editing fornite dal Child Extended BASIC permettono di correggere errori e di aggiungere e togliere elementi di programma. Se necessario è possibile cancellare l'intero programma. Tutte le possibilità di editing proprie del Child BASIC saranno discusse per esteso nel seguito del manuale. Qui ci limitiamo a descriverne brevemente gli aspetti fondamentali.

a. Correzione dei singoli caratteri. Se in una linea ancora in corso di battitura l'operatore rileva un carattere errato, può correggerlo immediatamente con il tasto RUBOUT. Ogni pressione del tasto elimina il carattere immediatamente precednte. Se non c'è nessun carattere precedente viene inviato un ritorno carrello ed è iniziata una nuova linea. Una volta che i caratteri indesiderati sono stati rimossi, possono essere rimpiazzati semplicemente battendo il resto della linea nella maniera desiderata.

Quando è battuto il RUBOUT, sono stampati una barra contraria (\) e, di seguito il caratter da cancellare. Ogni successiva pressione del tasto RUBOUT provoca la stampa del prossimo carattere da cancellare. La pressione di un nuovo carattere fa sì che sia stampata una nuova barra contraria ed il nuovo carattere. A questo punto tutti i caratteri che si trovano fra le due barre sono stati eliminati. Esempio:

```
100 X=\=X\Y=10
```

Premendo due volte consecutivamente il RUBOUT si elimina prima il segno "=", poi la "X". I due caratteri sono successivamente rimpiazzati da "Y=".

b. Correzione delle linee. Una linea in fase di stesura è eliminata battendo il carattere alfetta (@) invece del ritorno carrello. Questo è stampato automaticamente dopo che la linea è eliminata. Il medesimo risultato è ottenuto tramite il carattere Control/U.

Battere Control/A in luogo del ritorno carrello permette di usufruire di tutte le possibilità del comando EDIT (con esclusione del comando A) sulla linea attualmente in preparazione (v. par. 5-4).

c. Correzione di programmi. Il comando NEW permette di cancellare l'intero programma e tutte le variabili in memoria al momento. Il comando NEW è generalmente usato per ripulire tutta la memoria in vista dell'immissione di un nuovo programma.

II - ESPRESSIONI E FRASI

2-1 Espressioni

Le più semplici espressioni BASIC sono le singole costanti, variabili e funzioni.

a. Costanti. L'interprete BASIC accetta come costanti numeri interi e numeri reali in virgola mobile. Parimenti accetta costanti alfanumeriche (v. Cap. IV - Stringhe). I seguenti sono esempi di costanti numeriche valide:

343

5.567

Ø.Ø7

3.25E+Ø4

I dati immessi da terminale o le costanti numeriche di un programma possono essere composti da un numero qualunque di cifre fino alla lunghezza massima della linea, tuttavia solo le prime 17 sono significative.

Il formato di stampa di un numero è determi-

nato dalle regole seguenti:

1. Se il numero è negativo viene stampato un segno meno (-) alla sua sinistra. Se il numero è positivo, viene stampato uno spazio.

2. Se il valore assoluto del numero è un intero nel campo da Ø a 999999, esso viene stampato come un intero.

3. Se il valore assoluto del numero è maggiore o uguale a 9999999999999999 , questo viene stampato in virgola fissa senza alcun esponente.

4. Se il numero non rientra nelle categorie 1,2 e 3, viene usata la notazione scientifica.

I formati della notazione scientifica sono i seguenti:

SX.XXXXXESTT	singola precisione
SX.XXXXXXXXXXXXXXDSTT	doppia precisione

dove S rappresenta il segno della mantissa e dell'esponente, X le cifre della mantissa e T quelle dell'esponente. E e D si leggono "...per dieci elevato a...". Gli zero non significativi vengono soppressi

nella mantissa, mentre per l'esponente sono sempre stampate due cifre. Inoltre l'esponente deve essere compreso fra -38 e +38.

Il numero maggiore rappresentabile nel Child BASIC è 1.70141E38, il più piccolo numero positivo è invece 2.9387E-38.

Il Child Extended BASIC permette di rappresentare numeri in forma intera, in precisione singola e doppia. La classe di appartenenza di una costanta numerica è determinata in accordo alle regole seguenti:

1. Una costante con più di 7 cifre o 'D' al posto di 'E' nell'esponente è in doppia precisione.
2. Una costante non compresa fra -32768 e 32767 con 7 o meno cifre ed un punto decimale o con un 'E' nell'esponente è in precisione singola.
3. Una costante compresa fra -32768 e 32767 senza punto decimale è intera.
4. Una costante seguita da un punto esclamativo (!) è in precisione singola; una costante seguita dal segno di numero (#) è in precisione doppia.

Nel Child BASIC sono permessi due ulteriori tipi di costanti. Il primo è quello delle costanti esadecimali (rappresentazione numerica in base 16). Queste costanti sono esplicitamente designate anteponendo al numero il simbolo &H. Esse non possono contenere caratteri diversi dai numeri da Ø a 9 e dalle cifre da A ad F, o verrà segnalato un SYNTAX ERROR. Si hanno poi le costanti ottali (rappresentazione numerica in base 8) che vengono designate dal simbolo &O, od anche dal solo segno &.

In tutti i formati dopo il numero viene stampato uno spazio. Il BASIC inoltre controlla, prima di stampare un numero, se questo è contenuto per intero nella linea corrente. Se non lo è, invia un ritorno carrello e stampa per intero il numero sulla linea successiva.

b. Variabili.

1. Una variabile rappresenta simbolicamente un qualunque numero che venga assegnato ad essa. Il valore di una variabile può essere assegnato esplicitamente dal programmatore, oppure può essere il

risultato di certi calcoli all'interno di un programma. Prima che ad una variabile sia assegnato un valore ben definito, questo viene assunto uguale a zero.

Un nome di variabile può essere di qualsiasi lunghezza, ma ogni carattere alfanumerico dopo i primi due viene ignorato. Il primo carattere deve necessariamente essere una lettera. Si noti che nessuna parola riservata può apparire come nome di variabile, o come parte di nome di variabile. Di seguito riportiamo alcuni esempi di variabili BASIC, legali ed illegali:

Legali	Illegali
A	%A (il primo carattere non è alfabetico)
Z1	TO (è una parola riservata)
TP	
PTGT$	RGOTO (GOTO è riservata)

Ricordiamo che una variabile può essere anche

alfanumerica. Questa importante particolarità sarà discussa nel Cap. IV.

2. Il Child Extended BASIC consente l'uso di variabili intere e in doppia precisione, così come di variabili alfanumeriche ed in singola precisione. Il tipo di variabile può essere dichiarato esplicitamente usando uno dei simboli sottoriportati come ultimo carattere del nome di variabile:

Tipo	Simbolo
Stringa (da Ø a 255 caratteri)	$
Intero (da -32768 a 32767)	%
Precisione singola (fino a 7 cifre, esponente fra -38 e +38)	!
Precisione doppia (fino a 16 cifre, esponente fra -38 e +38)	#

Se il tipo di variabile non è dichiarato esplicitamente, viene determinato in base alla prima lettera del nome di variabile in accordo alla tavola dei tipi. Questa tavola può essere modificata tramite le seguenti frasi:

DEFINT r	Intera
DEFSTR r	Alfanumerica
DEFSNG r	Precisione singola
DEFDBL r	Precisione doppia

dove "r" è una lettera o un insieme di lettere, come mostriamo nel seguente esempio:

10 DEFINT I-N	Le variabili i cui nomi cominciano con le lettere da I a N devono essere di tipo intero.
20 DEFDBL D	Le variabili i cui nomi cominciano con la lettera D devono essere in doppia precisione.

Se non incontra frasi di definizione di tipo di variabile, il BASIC procede come se avesse eseguito una istruzione DEFSNG A-Z.

3. Dovunque è possibile, dovrebbero sempre essere usate variabili intere, sia perchè esse occupano il minor spazio in memoria, sia perchè l'aritmetica intera è molto più veloce delle altre.

c. Variabili con indice. Matrici e Vettori.

Spesso è vantaggioso riferirsi con lo stesso nome a diverse variabili. Ad esempio nel calcolo matriciale l'elaboratore tratta separatamente ogni elemento della matrice, ma per il programmatore è utile riferirsi all'intera matrice come un unico ente matematico. A tale scopo il Child BASIC mette a disposizione dell'utente le variabili con indice, o matrici. Il formato di una tale variabile è il seguente:

VV('indice'°,'indice'...°)

dove VV è un nome di variabile e gli indici sono espressioni intere. Gli indici possono essere racchiusi fra parentesi tonde o quadre. Il numero di dimensioni di una variabile con indice è arbitrario, limitato solo dalla lunghezza di una singola linea. L'indice più piccolo è zero.

Vediamo ora alcuni esempi:

A(5) Indica il sesto elemento della matrice unidimensionale (vettore) A. Il primo elemento è A(Ø).

MAT(I,2*J) L'indirizzo di questo elemento all'interno di una matrice bidimensionale è determinato dal valore dell'espressione in parentesi al momento in cui si fa riferimento alla variabile. Se un'espressione assume un valore decimale, questo viene troncato all'intero immediatamente minore. Se, ad es., I=3 e I=2.4, l'elemento in causa è MAT(3,4).

La frase DIM assegna memoria alle matrici e pone al valore iniziale zero tutti i loro elementi. Il formato della frase è qua riportato:

DIM VV('indice'°,'indice'...°)

dove VV è un nome di variabile consentito. L'indice è un'espressione intera, che specifica il massimo indice possibile per quella dimensione. Si noti che ogni frase DIM può essere applicata a più di una variabile. Ecco di seguito alcuni esempi:

```
110 DIM A(3),D$(2,2,2)
```

```
115 DIM R3%(4),B(10)
120 DIM P1(N),Z(2+I)
```

Come si può notare dall'ultimo esempio, le matrici possono essere dimensionate dinamicamente durante l'esecuzione del programma. Al momento in cui è eseguita la frase DIM, vengono calcolate le espressioni in parentesi ed i risulati sono tronacati al valore intero immediatamente minore.

Se al momento in cui in un programma si incontra una matrice non è stata eseguita nessuna frase DIM, il BASIC assume automaticamente che la matrice abbia un indice massimo pari a 10 (cioè 11 elementi) per ogni dimensione riscontrata nel riferimento a quella variabile. Se poi c'è un tentativo di far riferimento ad un elemento di una matrice, che è fuori dallo spazio assegnato nella frase DIM associata, viene inviato un messaggio di errore BS o SUBSCRIPT OUT OF RANGE (indice fuori campo). Questo succede ad esempio, quando è usato un numero errato di dimensioni nel far riferimento ad una matrice, come è mostrato nell'esempio che segue:

```
30 LET A(1,2,3)=X
```

quando A era stata dimensionata dalla frase

```
10 DIM A(2,2)
```

Si noti infine che si incorre in un errore di tipo REDIMENSIONED ARRAY, quando viene trovata una frase DIM per una matrice già dimensionata in precedenza. Questo succede spesso, quando una frase DIM appare dopo che ad una matrice è stata assegnata la dimensione 10 per mancanza di dimensionamento precedente.

d. Operatori e Precedenze.

Il Child Extended BASIC mette a disposizione del programmatore una vasta gamma di operatori logici ed aritmetici. L'ordine di esecuzione delle operazioni in un'espressione segue le regole di predenza riportate nella tabella seguente. Tale ordine può comunque essere specificato esplicitamente tramite l'uso delle parentesi secondo le normali regole algebriche. Gli operatori sono riportati nella tabella in ordine decrescente di precedenza. Operatori, che si trovano allo stesso livello della ta-

bella, hanno la stessa precedenza ed in tal caso l'ordine di esecuzione in un'espressione è quello da sinistra a destra.

Tabella delle Precedenze

1. Espressioni racchiuse fra parentesi ()
2. ^ elevamento a potenza. Ogni numero elevato a zero da 1. Zero elevato ad una potenza negativa dà un errore di tipo DIVISION BY ZERO.
3. - negazione, l'operatore monadico meno.
4. *,/ moltiplicazione e divisione.
5. \ divisione intera (v. par. 5-2).
6. MOD (v. par.5-2)
7. +,+ addizione e sottrazione
8. operatori di relazione
   - = uguale a
   - <> diverso da
   - < minore di
   - > maggiore di
   - <=, =< minore di o uguale a
   - >=, => maggiore di o uguale a

9. NOT logico, negazione

10. AND " , disgiunzione

11. OR " , unione

12. XOR " , OR esclusivo

13. EQV " , equivalenza

14. IMP " , implicazione

Si ricorda che gli operatori di relazione possono essere usati in qualsiasi espressione, e non solo in una frase IF. Le espressioni di relazione possono assumere solo il valore Vero (-1) o Falso (Ø).

e. Operatori Logici.

Gli operatori logici vengono usati per la manipolazione di bit e per le funzioni di algebra Booleana. Gli operatori AND, OR, NOT, XOR, EQV e IMP convertono i loro argomenti in numeri interi a 16 bit, nel campo -32768÷32767, rappresentati in aritmetica binaria con segno, in complemento a due.Dopo che le operazioni sono state eseguite, i risultati vengono restituiti nella medesima forma e nel medesimo campo. Se gli argomenti sono al di fuori di questo campo, viene stampata la segnalazione di errore

ILLEGAL FUNCTION CALL e l'esecuzione è arrestata. Qua sotto sono riportate le tavole della verità per gli operatori logici. Le operazioni vengono eseguite bit a bit, ossia sono esaminati i bit corrispondenti di ogni argomento ed il risultato è calcolato un bit alla volta.

AND

X	Y	X AND Y
Ø	Ø	Ø
Ø	1	Ø
1	Ø	Ø
1	1	1

OR

X	Y	X OR Y
Ø	Ø	Ø
Ø	1	1
1	Ø	1
1	1	1

NOT

X	NOT X
Ø	1
1	Ø

XOR

X	Y	X XOR Y
Ø	Ø	Ø
Ø	1	1
1	Ø	1
1	1	Ø

EQV

X	Y	X EQV Y
Ø	Ø	1
Ø	1	Ø
1	Ø	Ø
1	1	1

IMP

X	Y	X IMP Y
Ø	Ø	1
Ø	1	1
1	Ø	Ø
1	1	1

Vediamo alcuni esempi per chiarire come lavorano gli operatori logici. In questi esempi useremo la notazione NN_b per indicare la rappresentazione binaria di un numero:

63 AND 16=16	$63=111111_b$ e $16=10000$, per cui 63 AND 16=16
15 AND 14=14	$15=1111_b$ e $14=1110_b$, e quindi 15 AND 14=14
-1 AND 8=8	$-1=1111111111111111_b$ e $8=1000_b$, per cui -1 AND 8=8
4 OR 2=6	$4=100_b$ e $2=10_b$, da cui si ricava 4 OR 2=6
-1 OR -2=-1	$-1=1111111111111111_b$ e $-2=1111111111111110_b$, per cui -1 OR -2=-1
NOT 0=-1	il complemento bit a bit di 16 zero sono 16 uno, che sono la rappresentazione in complemento a due di -1

Una tipica applicazione degli operatori logici è quella di testare un numero binario al fine di individuare una determinata configurazione di bit.

Tali numeri potrebbero provenire dalle porte di ingresso del computer e riflettere quindi lo stato di una qualche periferica. Ulteriori applicazioni degli operatori logici saranno considerate nella discussione della frase IF.

f. La frase LET.

La frase LET è adoperata per assegnare un valore ad una variabile. Il formato della frase è il seguente:

LET'VV'='espressione'

dove VV è il nome di variabile e l'espressione è una qualsiasi espressione aritmetica, o logica od alfanumerica. Esempi:

```
100 LET V=X
110 LET I=I+1
```

Si noti che la parola LET, nella frase è opzionale, cosicchè un'espressione algebrica come:

```
120 V=2.4*(X+8)
```

è una frase di assegnazione legale.

2-17

Prima di concludere questo paragrafo ricordiamo che il BASIC, quando rileva un formato errato, caratteri illegali su una linea, punteggiatura inesatta o parentesi omesse stampa la segnalazione di errore SYNTAX ERROR. Si ha invece il messaggio di errore OVERFLOW, quando il risultato di una operazione è troppo grande per essere rappresentato nel formato numerico del Child Extended BASIC. In particolare i numeri positivi devono essere compresi fra 1E-38 e 1.70141E38, ed i negativi fra -1E-38 e -1.70141 E38. Infine un tentativo di divisione per zero provoca la segnalazione di errore DIVISION BY ZERO.

2-2. Salti, Cicli e Sottoprogrammi

a. Salti.

Oltre all'esecuzione sequenziale di linee di programma il BASIC consente di cambiare l'ordine di esecuzione delle istruzioni. Le frasi BASIC di salto sono GOTO, IF...THEN, ON...GOTO.

1. GOTO è un salto di tipo incondizionato. Il formato della frase è:

GOTO'nnnnn'

Dopo l'esecuzione della frase GOTO l'esecuzione del programma riprende 'incondizionatamente' dalla linea numerata nnnnn.

2. IF...THEN è un tipo di salto detto condizionato. Il suo formato è:

IF'espressione'THEN...GOTO'nnnnn'

dove l'espressione è una qualsiasi espressione logica, o aritmetica o di relazione valida, ed nnnnn è un numer di linea. Se il valore dell'espressione è diverso da zero, il BASIC salta alla linea nnnnn, altrimenti l'esecuzione riprende dalla immediatamente successiva alla frase IF...THEN.

Una forma alternativa della frase IF...THEN è la seguente:

IF'espressione'THEN'frase'

dove frase sta per una qualsiasi frase BASIC.

Vediamo alcuni esempi, che chiariranno meglio il funzionamento della frase:

10 IF A=10 THEN 40 Se l'espressione A=10 è vera (cioè il suo valore è -1, diverso da zero), il BASIC salta alla linea 40, altrimenti l'esecuzione riprende alla linea successiva.

15 IF A=B+C OR X THEN 100 Viene calcolata l'espressione dopo l'IF,e, se il suo valore non è zero, si salta alla 100. Altrimenti l'esecuzione continua alla linea successiva.

20 IF X THEN 5 Se X è diverso da zero, si salta alla 5.

25 IF X=Y THEN PRINT X Se l'espressione X=Y è vera, viene eseguita la frase PRINT; se è falsa la frase PRINT è tralasciata. In ogni caso l'esecuzione continua alla linea successiva.

35 IF X=Y+3 GOTO 40 E' equivalente alla frase IF...THEN corrispondente, eccetto che GOTO deve essere seguito obbligatoriamente da un numero di linea e non da un'altra frase.

2-20

Un ulteriore forma della frase IF...THEN è la seguente:

IF'espressione' THEN'YY'ELSE'ZZ'

dove YY e ZZ sono numeri di linea o frasi BASIC. Così l'istruzione :

10 IF X=2+Y THEN 30 ELSE PRINT "ERROR"

provoca il salto del programma alla linea 30, o l'esecuzione della frase PRINT, a seconda che l'espressione X=2+Y sia vera o falsa.

Il Child BASIC permette di annidare sulla medesima linea più frasi IF. Il livello di annidamento è limitato solo dalla lunghezza della linea. Così, ad esempio le frasi:

```
IF X>Y THEN PRINT "MAGGIORE" ELSE IF Y>X THEN
        PRINT "MINORE" ELSE PRINT "UGUALE"
```

e

```
IF X=Y THEN IF Y>Z THEN PRINT "X>Z" ELSE PRINT
        "Y<=Z" ELSE PRINT "X<>Y"
```

sono legali. Se una linea non contiene lo stesso numero di proposizioni ELSE e THEN, ogni ELSE è accop-

piato con il più vicino THEN non accoppiato. Perciò l'esecuzione dell'istruzione:

```
IF A=B THEN IF B=C THEN PRINT "A=C" ELSE
       PRINT "A<>C"
```

non provocherà la stampa "A<>C", se A<>B.

3. ON...GOTO fornisce un'ulteriore possibilità di salto condizionato. Il formato della frase è qua riportato:

ON'espressione'GOTO'lista di numeri di linea'

Dopo che il valore dell'espressione è troncato ad un valore intero, diciamo I, la frase provoca un salto al numero di linea che occupa nella lista la I-ma posizione. La frase può essere seguita da tanti numeri di linea, quanti ne entrano su una singola linea.

```
100 ON I GOTO 10,20,30,40      Salta alla linea
          indicata dall'I-mo numero dopo il GO-
          TO. Ossia:
          IF I=1 THEN GOTO 10
          IF I=2 THEN GOTO 20
          IF I=3 THEN GOTO 30
          IF I=4 THEN GOTO 40
```

Se I è uguale a zero, o se tenta di selezionare una linea non esistente (in questo caso I=>5), l'esecuzione continua dalla linea successiva alla frase ON...GOTO. Inoltre I non deve essere minore di zero o maggiore di 255, altrimenti si incorre nell'errore ILLEGAL FUNCTION CALL.

b. Cicli.

E' spesso desiderabile eseguire gli stessi calcoli o su dati differenti o ripetitivamente sugli stessi dati. Questo scopo si raggiunge tramite le frasi FOR e NEXT. Il formato della frase FOR è il seguente:

FOR 'variabile'='X'TO'Y'°STEP'Z'°

dove X, Y e Z sono espressioni. Quando la frase FOR è incontrata per la prima volta, vengono calcolate le espressioni. Alla variabile viene assegnato il valore di X, detto valore iniziale. Poi il BASIC esegue le frasi successive alla FOR nel modo usuale. Quando incontra una frase NEXT, il passo (STEP) Z viene sommato alla variabile che è poi confrontata con il valore finale Y. Se il passo è positivo e la

variabile è minore o uguale al valore finale, o se il passo è negativo e la variabile è maggiore o uguale al valore finale, il BASIC torna alla frase immediatamente seguente la FOR. In caso contrario l'esecuzione procede con la frase successiva alla NEXT. Se il passo non è specificato, viene assunto uguale a 1.

Esempi:

```
10 FOR I=2 TO 11
```

Il ciclo è eseguito 10 con la variabile I che assume tutti i valori interi fra 2 e 11.

```
20 FOR V=1 TO 9.3
```

Questo ciclo sarà eseguito 9 volte, finchè V supera il valore 9.3.

```
30 FOR V=10*N TO 3.4/Z STEP SQR(R)
```

Le espressioni del valore iniziale e finale e dello step, non necessariamente intere, vengono calcolate una volta per

tutte prima che il ciclo abbia inizio.

```
40 FOR V=9 TO 1 STEP -1
```

Questo ciclo è eseguito nove volte.

I cicli FOR...NEXT possono essere annidati, cioè il BASIC è in grado di eseguire un ciclo FOR... ...NEXT all'interno di un altro ciclo. Un esempio di due ciccli annidati è il seguente:

```
100 FOR I=1 TO 10
120 FOR J=1 TO I
130 PRINT A(I,J)
140 NEXT J
150 NEXT I
```

La linea 130 stampa 1 elemento di A per I=1, 2 per I=2 e così via. Nel caso di cicli annidati i nomi delle variabili di ciclo devono essere differenti. La frase NEXT per la variabile di ciclo interno (nell'esempio J) deve precedere quella per la variabile esterna (I). Il numero di livelli di annidamento è limitato solo dalla disponibilità di memoria utente.

Il formato della frase NEXT è il seguente:

NEXT°'variabile'°,'variabile'...°°

dove ogni termine 'variabile' sta per la variabile di ciclo di una frase FOR di cui la NEXT è il punto di chiusura. Una frase NEXT senza variabile si accoppia con il FOR più recente. Nel caso di cicli annidati, che hanno lo stesso punto di fine, può essere usata per tutti una sola frase NEXT. In tal caso la prima variabile della lista deve essere quella del ciclo più recente, la seconda del penultimo e così via. Se il BASIC incontra una frase NEXT prima del FOR corrispondente, viene stampato il messaggio di errore NEXT WITHOUT FOR (ossia NEXT SENZA FOR) e l'esecuzione ha termine.

c. Sottoprgrammi.

Se la stessa operazione o serie di operazioni deve essere eseguita in più punti di un programma, la quantità di memoria richiesta ed il tempo di programmazione sono minimizzati mediante l'uso di sottoprogrammi. Un sottoprogramma è un insieme finito di istruzioni, che vengono eseguite secondo le regole usuali, dopo che si è saltato ad esso trami-

te la frase GOSUB (GO to SUBroutine). L'esecuzione di un sottoprogramma è terminata dall'istruzione RETURN, che salta indietro alla frase immediatamente seguente l'istruzione GOSUB, che per ultima ha provocato il salto a quel sottoprogramma. Il formato della frase GOSUB è:

GOSUB'numero di linea'

dove il numero di linea è quello della prima linea del sottoprogramma. Un sottoprogramma può essere richiamato da diversi punti di un programma, e può a sua volta richiamare un altro sottoprogramma. Tale annidamento di sottoprogrammi è limitato solo dalla disponibilità di memoria.

Il salto ai sottoprogrammi può essere anche di tipo condizionato tramite l'uso della frase ON... ...GOSUB il cui formato è:

ON'espressione'GOSUB'lista di numeri'

L'esecuzione di questa frase è analoga a quella della frase ON...GOTO eccetto che i numeri di linea sono quelli delle prime linee dei sottoprogrammi. L'esecuzione riprende dalla riga successiva al ON...GOSUB dopo il ritorno da uno dei sottoprogrammi speci-

ficati nella lista.

d. Errori di OUT OF MEMMORY

Mentre il BASIC non limita di suo l'annidamento di cicli, sottoprogrammi e salti, le limitazioni di memoria impongono restrizioni sulla grandezza e sulla complessità dei programmi. Quando un programma richiede più memoria di quanta ne sia disponibile, viene emessa la segnalazione di errore OUT OF MEMORY (letteralmente: fuori dalla memoria). Nelle Appendici al manuale è riportata una spiegazione sull'ammontare di memoria necessaria per girare i programmi.

2-3. Ingresso/Uscita

a. INPUT.

La frase INPUT (ingresso) provoca una richiesta di immissione di dati da terminale. Il suo formato è:

INPUT'lista di variabili'

L'effetto della frase INPUT è di assegnare alle va-

riabili specificate nella lista i valori battuti sul terminale dall'operatore. Quando il BASIC esegue una frase INPUT, sul terminale viene stampato un punto interrogativo per indicare una richiesta di informazioni necessarie per la prosecuzione del programma. L'operatore batte allora i numeri e/o le stringhe richieste, separando con una virgola ogni singolo termine, ed infine un ritorno carrello. Se il dato immesso non è valido (ad esempio una stringa invece di un numero) il BASIC stampa il messaggio 'REDO FROM START?' ed attende che siano battuti i dati corretti. Se erano stati richiesti dalla frase INPUT più dati di quanti ne vengono battuti, il BASIC stampa '??' e l'esecuzione attende i dati necessari. Se al contrario vengono battuti più dati di quante fossero le variabili nella lista, viene stampato l'avviso 'EXTRA IGNORED' e l'esecuzione prosegue. Dopo che sono stati immessi tutti i dati richiesti, l'esecuzione riprende normalmente dalla frase successiva all'INPUT.

Alla frase INPUT può essere aggiunta una stringa di avviso opzionale, che ha lo scopo di ricordare i dati da immettere. Il formato della frase è allora:

INPUT°"'stringa mnemonica'";°'lista'

In tal caso prima del punto interrogativo viene stampata la stringa, dopo di che tutte le operazioni procedono come visto in precedenza. La stringa mnemonica deve essere racchiusa fra virgolette (") e separata con un punto e virgola dalla lista delle variabili. Ad esempio l'esecuzione della linea:

```
100 INPUT "QUAL E' IL VALORE";X,Y
```

provoca l'uscita:

```
QUAL E' IL VALORE?
```

dopo di che l'elaboratore attende che l'utente immetta i valori di X e Y.

Si tenga infine presente che un ritorno carrello in risposta ad una frase INPUT provoca la ripresa dell'esecuzione con i valori delle variabili della lista lasciati inalterati.

b. PRINT.

La frase PRINT (stampa) provoca la stampa di dati sul terminale. La più semplice frase PRINT è:

```
PRINT
```

che stampa un ritorno carrello. L'effetto è quindi

quello di saltare una linea. Comunque la forma più comune della frase PRINT è:

PRINT'lista di espressioni'

che provoca la stampa delle espressioni specificate nella lista. Si ricorda che possono essere stampate stringhe alfanumeriche, purchè racchiuse fra virgolette (").

La posizione di stampa è determinata dalla punteggiatura usata per separare gli elementi della lista. Il Child BASIC divide la linea di stampa in zone costituite ciascuna da 14 spazi. Una virgola fa sì che la stampa del valore della successiva espressione cominci all'inizio della successiva zona di 14 colonne. Invece un punto e virgola fa sì che la stampa inizi immediatamente dopo quella dell'ultimo valore. Se la lista di espressioni termina con una virgola o con un punto e virgola, una successiva frase PRINT comincia a stampare sulla medesima linea secondo le modalità viste sopra. Altrimenti viene stampato un ritorno carrello.

c. DATA, READ, RESTORE

1. La frase DATA. I dati numerici o alfanume-

rici necessari in un programma possono essere scritti direttamente nelle frasi stesse del programma, oppure immessi da dispositivi periferici, o infine essere letti dalle frasi DATA. Il formato di una tale frase è:

DATA'lista'

dove gli elementi della lista sono costanti numeriche o alfanumeriche separate da virgole. L'effetto della frase è di memorizzare in forma codificata i valori della lista per avervi poi accesso tramite la frase READ (v. sotto). Esempi:

```
10 DATA 3,5,-7E3,.035
20 DATA " ECCO",CHILD
```

Si noti che gli spazi iniziali e finali nelle stringhe alfanumeriche vengono soppressi a meno che le stringhe non siano racchiuse fra virgolette.

2. La frase READ. Ai dati memorizzati tramite la frase DATA si ha accesso attraverso la frase READ, il cui formato è:

READ'lista di variabili'

dove gli elementi della lista sono nomi di variabili

separati da virgole. L'effetto della READ è di assegnare i valori nelle liste delle frasi DATA alle variabili corrispondenti nella lista della frase READ. Questa operazione viene effettuata elemento per elemento, da sinistra a destra, finchè la lista della frase READ è esaurita. Se vi sono nella lista della READ più nomi di variabili di quanti sono complessivamente i valori di tutte le frasi DATA, viene stampato il messaggio di errore OUT OF DATA. Se in una frase DATA vi sono più valori di quanti ne sono letti da una singola frase READ,la successiva READ che viene eseguita comincia a leggere dal primo elemento non letto della DATA. Una singola frase READ può riferirsi a più di una frase DATA, così come diverse READ possono accedere agli elementi di un'unica DATA.

Si tenga presente che da un errato formato della lista di una frase DATA può derivare una segnalazione SYNTAX ERROR. In tal caso il numero di linea nel messaggio di errore si riferisce alla linea della frase DATA in cui si è verificato l'errore.

3. La frase RESTORE. Dopo l'esecuzione di que-

sta istruzione i dati a cui accede una frase READ sono quelli a partire dal primo elemento della prima frase DATA. Ciò quindi permette una rilettura dei dati.

d. CSAVE*, CLOAD*.

Matrici numeriche possono essere memorizzate su cassetta magnetica (CSAVE*) o caricate da cassetta (CLOAD*), usando appunte le frasi CSAVE* e CLOAD*. I formati delle due frasi sono:

CSAVE*'nome di matrice'

e

CLOAD*'nome di matrice'.

In fase di caricamento su nastro alla matrice sono fatti precedere quattro bytes, aventi la configurazione binaria 10001000. Questi bytes vengono ricercati al momento in cui la matrice deve essere riletta in memoria centrale tramite la frase CLOAD*. Perciò il numero degli elementi scritti in totale per ogni matrice è quattro più:

8*'numero degli elementi'

4*'numero degli elementi'

2*'numero degli elementi'

a seconda che la matrice sia rispettivamente in precisione doppia, singola o intera.

Quando una matrice viene trasferita su nastro, i suoi elementi vengono scritti con l'indice più a sinistra che varia più velocemente, poi il secondo indice da sinistra e così via. Perciò:

```
DIM A(10)
.........
CSAVE*A
```

scrive nell'ordine A(0),A(1),...,A(10), mentre:

```
DIM A(10,10)
............
CSAVE*A
```

scrive A(0,0),A(1,0),...A(10,0),A(0,1),...,A(10,1), ...A(0,10),...,A(10,10).

Usando questa particolarità, è possibile scrivere una matrice come bidimensionale e rileggerla come unidimensionale.

Infine si presti bene attenzione a non memorizzare su nastro una matrice in doppia precisione, e rileggerla come matrice in precisione singola o addirittura intera. Questo porterebbe certamente ad

avere dei valori privi di utilità.

e. Miscellanee.

1. Lo stato dei ports di ingresso dell'elaboratore può essere tenuto sotto controllo tramite il comando WAIT, che ha il seguente formato:

WAIT'I,J'°,'K'°

dove I è il numero del port da controllare, e J e K sono espressioni intere. In seguito a questa istruzione viene operato un OR esclusivo fra lo stato del port e K, dopo di che viene eseguito un AND fra il risultato ottenuto e J. L'esecuzione del programma è sospesa, fintanto che il risultato delle operazioni sopra descritte è diverso da zero. In pratica J preleva i bits del port I che devono essere testati, e l'esecuzione è sospesa, fintanto che quei bits differiscono dai corrispondenti bits di K. Quando questi coincidono l'esecuzione del programma riprende dalla frase immediatamente successiva il comando WAIT. Nel caso che K non sia specificato, esso viene assunto uguale a zero. I,J e K devono essere compresi fra Ø e 255.

Esempi:

WAIT 20,6 L'esecuzione si arresta, finchè il bit 1 o il bit 2 (o entrambi) del port 20 sono uguali a 1.

WAIT 10,255,7 L'esecuzione si arresta, fintanto che sul port 10 non si verifica la configurazione binaria 00000111.

2. POKE, PEEK. I dati possono essere immessi in memoria in forma binaria con l'uso della frase POKE, il cui formato è:

POKE'I,J'

dove I e J sono espressioni intere. La frase POKE memorizza il byte J nella locazione specificata dal valore di I. Questo deve essere compreso fra 0 e 65536, mentre J può variare fra 0 e 255.

Bisogna ovviamente prestare attenzione a non memorizzare dei dati nell'aria di memoria riservata al Child BASIC, che potrebbe essere irrimediabilmente distrutto, tanto da richiedere un nuovo caricamento dell'interprete.

La funzione complementare di POKE è PEEK, il cui formato è il seguente:

PEEK('I')

dove I è un'espressione intera, il cui valore, che deve sempre essere compreso fra Ø e 65536, specifica la locazione di memoria da cui deve essere letto un dato. Il valore che si ottiene in seguito all'esecuzione di questa frase è un intero compreso fra Ø e 255.

Le frasi PEEK e POKE risultano particolarmente utili per scambiare argomenti e risultati fra il programma principale scritto in BASIC e sottoprogrammi in linguaggio macchina.

3. OUT, INP. Il formato della frase OUT è il seguente:

OUT 'I,J'

dove I e J sono espressioni intere. La frase OUT emette il byte espresso da J sul port di uscita I. Sia I che J devono essere compresi fra Ø e 255.

Infine la funzione INP è chiamata nel modo seguente:

INP('I')

2-38

L'effetto della funzione INP è quello di leggere un byte dal port di ingresso I, dove I è, al solito, un'espressione intera, che può variare fra Ø e 255. Un esempio dell'uso di tale funzione è il seguente:

```
1Ø IF INP(J)=25 THEN PRINT "SI'".
```

III - FUNZIONI

Il Child BASIC adoota per le funzioni la consueta notazione matematica. Il formato di una chiamata di funzione è quindi il seguente:

'nome'('argomento'°,'argomento'...°)

dove il nome è quello di una funzione definita in precedenza e gli argomenti sono una o più espressioni, separate da virgole. Le chiamate di funzioni possono a loro volta far parte di espressioni, cosicchè frasi come

```
1Ø LET T=(F*SIN(T))/P
2Ø C=SQR(A^2+B^2+2*A*B*COS(T))
```

sono perfettamente legali.

3-1 Funzioni Intrinseche

Il Child BASIC mette a disposizione dell'utendiverse funzioni di uso frequente, che possono essere richiamate da qualsiasi programma senza bisogno di ulteriori definizioni. Al par. 6-3 è riportata una lista

delle funzioni intrinseche disponibili.

3-2 Funzioni Definite dall'Utente

a. La frase DEF. Il programmatore può definire funzioni che non sono incluse nella lista delle funzioni intrinseche per mezzo della frase DEF, il cui formato è qui riportato:

DEF'nome'('lista di variabili')='espressione'

dove il nome della funzione deve essere FN seguito da un nome di variabile consentito e gli elementi della lista di variabili sono nomi di variabili "mute". Le variabili mute rappresentano gli argomenti nella chiamata di funzione. Si noti che in una funzione definita dall'utente il numero degli argomenti è arbitrario, e che sul lato destro dell'equazione può apparire una espressione qualsiasi, purchè contenuta in una unica linea. Inoltre le funzioni possono essere di qualsiasi tipo, anche alfanumeriche. Esempi:

```
10 DEF FNAVE(V,W)=(V+W)/2
11 DEF FNCON$(V$,W$)=RIGHT$(V$+W$,5)
```

```
12 DEF FNRAD(DEG)=3.14159/180*DEG
```

Una chiamata alla funzione definita alla linea 11, fa sì che sia eseguita la concatenazione di V$ e W$, dopo di che vengono ritornati alla frase chiamante i cinque caratteri più a destra risultanti da tale operazione. Per quanto riguarda poi la funzione definita alla linea 11, essa, quando viene chiamata con il valore di un angolo in gradi, risponde restituendo il valore equivalente in radianti.

Una funzione può essere ridefinita eseguendo un'altra frase DEF con lo stesso nome. Ovviamente una frase DEF deve essere eseguita, prima che sia chiamata la funzione che essa definisce.

b. USR. La funzione USR permette di richiamare sottoprogrammi scritti in linguaggio macchina.Per questa importantissima particolarità del Child BASIC si veda l'Appendice E.

3-3 Errori

Quando viene effettuata in modo improprio una

chiamata di funzione, il BASIC stampa il messaggio di errore ILLEGAL FUNCTION CALL. Alcune possibili cause del verificarsi della suddetta situazione sono le seguenti:

1. Un indice di matrice negativo. Ad esempio A(-1).
2. Un indice di matrice troppo grande (>32767).
3. Argomento negativo o nullo per LOG.
4. Argomento negativo per SQR.
5. A^B con A negativo e B non intero.
6. Una chiamata a USR senza che nessun sottoprogramma in linguaggio macchina sia stato caricato.
7. Argomenti impropri per MID$, LEFT$, RIGHT$, INP, OUT, WAIT, PEEK, POKE, TAB, SPC, INSTR, STRING$, SPACE$ o ON...GOTO.

Un tentativo di chiamare una funzione utente, che non è precedentemente apparsa in una frase DEF, provoca l'errore UNDEFINED USER FUNCTION.

Infine si verifica l'errore TYPE MISMATCH, se ad una funzione che attende un argomento alfanumerico viene passato un valore numerico o viceversa.

IV - STRINGHE

Nel Child Extended BASIC le espressioni possono sia avere valori numerici, sia essere costituite da stringhe di caratteri. Il Child BASIC fornisce un vasto ed interessante insieme di frasi e funzioni per la manipolazione di stringhe alfanumeriche. Molte frasi sono già state discusse, cosicchè in questo capitolo sarà trattata solo la loro particolare applicazione al caso delle stringhe.

4-1 Dati Alfanumerici

Una stringa è definita come una lista di caratteri alfanumerici in numero compreso fra Ø e 255. Le stringhe possono essere tanto delle costanti che delle variabili. Le costanti alfanumeriche sono delimitate da virgolette (") poste all'inizio ed alla fine. I nomi di variabili alfanumeriche terminano con il simbolo del dollaro ($). Vediamo alcuni esempi:

A$="ABBC" Assegna alla variabile A$ la strin-

ga dei quattro caratteri "ABBC"

B9$="14A/56" Assegna alla variabile B9$ la stringa "14A/56"

FOCFO$="D$" Assegna alla variabile FOCFO$ la stringa "D$"

Le stringhe battute in risposta ad una frase INPUT non devono essere necessariamente racchiuse fra virgolette.

Le matrici alfanumeriche possono essere dimensionate esattamente come ogni altro tipo di matrice tramite l'uso della frase DIM. Ogni elemento di una matrice alfanumerica è una stringa che può essere lunga al massimo 255 caratteri. Il numero totale di caratteri usati per stringhe alfanumeriche ad ogni passo del programma non deve superare l'ammontare totale di memoria assegnato alle stringhe, o verrà segnalato l'errore OUT OF STRING SPACE. Lo spazio riservato alle stringhe è assegnato tramite il comando CLEAR, che sarà esposto nel paragrafo 6-2.

4-2 Operazioni sulle Stringhe

a. Operatori di confronto. Gli operatori di confronto per le stringhe sono i medesimi visti per i numeri:

=	uguale a
<>	diverso da
<	minore di
>	maggiore di
=<,<=	minore di o uguale a
=>,>=	maggiore di o uguale a

Il confronto è effettuato carattere per carattere in base ai codici ASCII, finchè è trovata una differenza. Se durante il confronto viene raggiunta la fine di una stringa, tale stringa è considerata minore. In Appendice B sono riportati i codici ASCII.
Esempi:

A<Z	il codice ASCII di A è 41_H, di Z é $5A_H$.
" A">"A"	nelle stringhe alfanumeriche gli spazi sono significativi.

b. Espressioni alfanumeriche. Le espressioni alfanumeriche sono formate da costanti, variabili e funzioni alfanumeriche connesse dall'operatore di concatenazione (+). L'effetto di questo operatore è di aggiungere la stringa alla destra dell'operatore alla fine della stringa sulla sinistra. Se il risultato è una stringa maggiore di 255 caratteri, viene stampato il messaggio di errore STRING TOO LONG ed arrestata la esecuzione.

c. Ingresso/Uscita. Le medesime frasi usate per l'ingresso e l'uscita di normali dati numerici, possono essere usate altrettanto bene per dati alfanumerici.

1) INPUT, PRINT. Le frasi INPUT e PRINT leggono e scrivono stringhe sul terminale. Le stringhe non devono necessariamente essere racchiuse fra virgolette, ma se non lo sono, gli spazi iniziali vengono ignorati e la stringa è considerata conclusa non appena si incontra la prima virgola o i primi due punti. Esempi:

10 INPUT X$	Legge una stringa e la assegna alla variabile X$.
20 INPUT CRO$,DRO$	Legge due stringhe.
30 PRINT X$,"MA, VEDI"	Stampa due stringhe, inclusi, nella seconda, spazi e punteggiatura.

2) DATA, READ. Le frasi DATA e READ per dati alfanumerici agiscono nel medesimo modo visto per dati numerici. Per quanto riguarda le convenzioni sui formati vale quanto detto a pagina precedente per le frasi INPUT e PRINT.

4-3 Funzioni Alfanumeriche

Anche per quanto riguarda il formato di chiamata di funzioni intrinseche alfanumeriche, vale ciò che era stato visto a proposito delle funzioni numeriche. Nel par. 6-3 è riportata una lista comprendente tutte le funzioni alfanumeriche.

4-6

Come detto nei paragrafi precedenti l'utente ha la possibilità di definire speciali funzioni alfanumeriche tramite l'uso della frase DEF.

Le funzioni alfanumeriche devono terminare con il simbolo del dollaro ($).

V - ALCUNI IMPORTANTI ASPETTI DEL CHILD EXTENDED BASIC

Il Child Extended BASIC mette a disposizione dell'utente un certo numero di frasi, operatori, funzioni e comandi, che non sono normalmente disponibili in versioni di BASIC che girano in 4 od 8K. Al fine di mettere nel maggior risalto possibile la potenza di queste opzioni, esse sono state raggruppate insieme all'interno di questo unico capitolo.

5-1 Frasi

a. ERASE. La frase ERASE elimina le matrici da un programma e permette di utilizzare per altri fini lo spazio che queste occupavano in memoria. Il formato della frase è il seguente:

ERASE'lista di variabili con indice'

dove gli elementi della lista sono nomi legali di variabili matriciali separati da virgole. La frase ERA-

SE agisce esclusivamente sulle matrici e non sugli elementi di queste. Se in una lista appare un nome sconosciuto all'interno del programma, è segnalato l'errore ILLEGAL FUNCTION CALL. Le matrici cancellate in una frase ERASE possono in seguito essere ridimensionate, ma, ovviamente, i vecchi valori vengono perduti. Esempio:

```
10 DIM A(5,5)
.....
.....
60 ERASE A
70 DIM A(100)
```

b. LINE INPUT. Spesso è desiderabile assegnare un'intera linea in ingresso ad una variabile alfanumerica senza l'uso di virgolette ed altri delimitatori di campo. La frase LINE INPUT offre questa facilitazione. Il formato della frase è il seguente:

```
LINE INPUT°"'stringa d'avviso'",°;'nome di
                variabile alfanumerica'
```

La stringa d'avviso è un messaggio che viene stampato sul terminale prima che l'ingresso venga accettato. Al contrario di quanto visto per la frase INPUT, qui non viene stampato un punto interrogativo, a meno che questo non faccia parte della stringa d'avviso. Tutto ciò che viene immesso dalla fine del messaggio al ritorno carrello viene assegnato alla variabile alfanumerica specificata. Una frase LINE INPUT può essere elusa premendo Control/C. In tal caso il BASIC si porta al livello comandi e stampa OK, dopo di che l'esecuzione può essere ripresa battendo CONT.

Si noti che LINE INPUT distrugge il buffer d'ingresso, cosicchè non si può passare in fase di Edit con il comando Control/A per una riesecuzione della frase.

c. SWAP. La frase SWAP consente di scambiare fra di loro i valori di due variabili. Il suo formato è il seguente:

SWAP'variabile, variabile'

Il valore della seconda variabile è quindi assegnato alla prima e viceversa. Una o entrambe le variabili

possono essere elementi di una matrice. Nel caso che una o entrambe le variabili non sono elementi di matrice e non è stato loro assegnato nessun valore, si incorre nell'errore ILLEGAL FUNCTION CALL. Inoltre le due variabili devono essere del medesimo tipo, altrimenti viene segnalato l'errore TYPE MISMATCH.
Esempio:

```
10 INPUT P$,U$
20 SWAP P$,U$
30 PRINT P$,U$
RUN
?PRIMO,ULTIMO              Dati immessi
ULTIMO          PRIMO      Stampa del computer
```

d. TRON, TROFF. Per facilitare le operazioni di debug il Child BASIC mette a disposizione dell'utente due frasi che permettono di seguire la traccia dell'esecuzione di un programma. Quando la frase TRON attiva il flag "traccia", viene stampato il numero di ogni linea del programma che viene eseguita. I numeri di linea vengono evidenziati racchiudendoli fra paren-

tesi quadre. La funzione è disabilitata tramite l'esecuzione della frase TROFF.

Esempio:

```
TRON                          eseguita in modo diretto
OK                            stampato dal computer
10 PRINT 1:PRINT "A"          battuto dall'operatore
20 STOP
RUN
[10] 1                        numeri di linea e uscite stampati dall'elaboratore
A
[20]
BREAK IN 20
```

Si tenga infine presente che fra i vari effetti del comando NEW vi è anche quello di disattivare il flag "traccia".

e. IF...THEN...ELSE. Vedi par. 2-2.

f. DEFINT, DEFSNG, DEFDBL, DEFSTR. Vedi par. 2-1.

g. WIDTH. La frase WIDTH stabilisce la larghezza della linea di stampa sul terminale, espressa in caratteri. Il formato della frase è il seguente:

WIDTH 'espressione intera'

Esempi:

WIDTH 84

WIDTH 52

Il valore dell'espressione deve essere compreso fra 15 e 255, altrimenti si incorre nell'errore ILLEGAL FUNCTION CALL.

h. Trappole per errori. Il Child Extended BASIC dà all'utente la possibilità di scrivere routines per la rivelazione ed il trattamento degli errori, con le quali si può cercare o di porre rimedio agli errori o, comunque, di darne una spiegazione più completa di quanto non faccia il semplice messaggio emesso dall'interprete. Questa facilitazione è possibile grazie alle frasi ON ERROR GOTO, RESUME ed ERROR e alle variabili ERR ed ERL.

1) Come si appresta la trappola.

La frase ON ERROR GOTO specifica la linea del programma BASIC alla quale inizia il sottoprogramma di trattamento dell'errore. Il formato della frase è il seguente:

ON ERROR GOTO'numero di linea'

Ovviamente la frase ON ERROR GOTO deve essere eseguita prima del punto in cui l'operatore suppone che si verifichi l'errore. Una volta che la frase è stata eseguita, tutti gli errori individuati provocheranno l'inizio dell'esecuzione della routine specificata. Se il numero di linea specificato nella frase non esiste, viene stampato il messaggio di errore UNDEFINED LINE NUMBER.

2) Come si disinnesca la trappola.

L'esecuzione dell'istruzione 'ON ERROR GOTO Ø' disabilita la routine di cattura degli errori, cosicchè ogni errore successivo provocherà la stampa dell'usuale messaggio di errore da parte dell'interprete BASIC e l'arresto dell'esecuzione del programma. Se

in una routine "trappola" compare la frase 'ON ERROR GOTO Ø', il BASIC si arresterà e stamperà il messaggio relativo all'errore che ha causato lo scatto della trappola stessa. E' quindi opportuno che ogni programma "trappola" esegua una istruzione 'ON ERROR GOTO Ø', nel caso che si verifichi un errore, per cui non è stata prevista nessuna contromisura.

N.B. Se durante l'esecuzione di una routine "trappola" si verifica un errore, verrà stampato il consueto messaggio di errore e l'esecuzione arrestata. Le trappole per errori non catturano errori all'interno delle routine stesse.

3) Le variabili ERR ed ERL.

Quando la routine di trattamento dell'errore viene chiamata, la variabile ERR contiene il codice dell'errore che ha provocato la chiamata. Di seguito sono elencati i vari codici di errore ed i corrispondenti messaggi. Al par. 6-5 si trova poi una

dettagliata discussione sui singoli errori.

Codice	Errore
1	NEXT WITHOUT FOR
2	SYNTAX ERROR
3	RETURN WITHOUT GOSUB
4	OUT OF DATA
5	ILLEGAL FUNCTION CALL
6	OVERFLOW
7	OUT OF MEMORY
8	UNDEFINED LINE
9	SUBSCRIPT OUT OF RANGE
10	REDIMENSIONED ARRAY
11	DIVISION BY ZERO
12	ILLEGAL DIRECT
13	TYPE MISMATCH
14	OUT OF STRING SPACE
15	STRING TOO LONG
16	STRING FORMULA TOO COMPLEX
17	CAN'T CONTINUE
18	UNDEFINED USER FUNCTION
19	NO RESUME

Codice	Errore
20	MISSING OPERAND
21	RESUME WITHOUT ERROR
22	UNPRINTABLE ERROR
23	LINE BUFFER OVERFLOW

La variabile ERL contiene il numero della linea in cui è stato rilevato l'errore. Ad esempio se l'errore si è verificato alla linea 200, il valore di ERL sarà pari a 200. Se la frase che ha causato l'errore era una frase in modo diretto, ERL assume il valore 65535 decimale. Perciò per controllare se un'errore si è verificato durante il modo diretto di funzionamento, è necessario usare l'istruzione:

```
IF 65535=ERL THEN...
```

In ogni altro caso la frase dovrà essere del tipo:

```
IF ERL='numero di linea'THEN...
```

A questo proposito si faccia attenzione al fatto che, se il numero di linea si trova sulla sinistra dell'equazione,non può essere rinumerato tramite il RENUM.

4) La frase RESUME.

La frase RESUME è utilizzata per riprendere l'esecuzione del programma BASIC principale, dopo lo svolgimento della routine di trattamento dell'errore. Per far ciò l'utente ha tre possibilità. Egli può riprendere (to RESUME in inglese) l'esecuzione dalla frase che ha provocato l'errore, dalla successiva, o, infine, da una qualsiasi altra linea. L'istruzione da usare rispettivamente nei tre casi è:

- RESUME oppure RESUME 0
- RESUME NEXT
- RESUME'numero di linea'

In quest'ultimo caso 'numero di linea' deve essere ovviamente diverso da zero.

5) Un esempio.

Il seguente esempio mostra come funziona una semplice routine per il trattamento dell'errore.

```
100 ON ERROR GOTO 500
200 INPUT "I NUMERI DA DIVIDERE SONO";X,Y
```

```
210 Z=X/Y
220 PRINT "IL QUOZIENTE E'";Z
230 GOTO 200
500 IF ERR=11 AND ERL=210 THEN 520
510 ON ERROR GOTO 0
520 PRINT "NON SI DIVIDE UN NUMERO PER ZERO!"
520 RESUME 200
```

6) La frase ERROR.

Tramite la frase ERROR è possibile forzare il salto ad una routine "trappola". Il pricipale uso della frase ERROR è quello di permettere all'utente di definirsi i propri codici di errore, da trattare poi adeguatamente tramite un sottoprogramma "trappola" secondo le regole viste in precedenza. Il formato della frase è:

ERROR 'espressione intera'

Quando definisce i propri codici di errore l'operatore dovrebbe ovviamente scegliere dei codici maggiori di quelli già usati dall'interprete BASIC. Se la 'espressione intera' della frase ERROR non è compresa

fra Ø e 255, si incorre nell'errore ILLEGAL FUNCTION CALL. Naturalmenet la frase ERROR può essere utilizzata per forzare gli errori standard del Child BASIC, come ad esempio il SYNTAX ERROR. L'uso di una istruzione ERROR per forzare la stampa di un messaggio di errore, per il quale nessun testo è definito, provocherà l'uscita del messaggio UNPRINTABLE ERROR.

5-2 Operatori

Due sono gli operatori che mettiamo in risalto in questo capitolo.

a. Divisione intera. La divisione intera, indicata dal simbolo \ (barra contraria), forza i suoi argomenti in forma intera e tronca il quoziente ad un valore intero. In formula:

```
A\B=FIX(INT(A)/INT(B))
```

Per quanto riguarda la precedenza questo operatore viene subito dopo la moltiplicazione e la divisione in virgola mobile. Molto importante è il fatto che la divisio-

ne intera è circa otto volte più veloce dell'usuale divisione in virgola mobile.

b. Modulo Aritmetico - l'operatore MOD.

A MOD B fornisce il resto della divisione fra A e B. Più precisamente:

A MOD B=INT(A)-(INT(B)*(A\B))

Se B=Ø, si ha l'errore DIVISION BY ZERO. La precedenza dell'operatore MOD segue immediatamente quella della divisione intera.

5-3 Funzioni

a. Funzioni intrinseche. Per una lista completa delle funzioni intrinseche disponibili nel Child Extended BASIC e per una descrizione del loro uso rimandiamo al par. 6-3.

b. La frase DEFUSR. Fino a dieci sottoprogrammi in linguaggio macchina possono essere definiti tramite la frase DEFUSR il cui formato è:

DEFUSR°'cifra da Ø a 9'°='espressione intera'

Possibili esempi sono:

DEFUSR1=&100000

DEFUSR3=27196

DEFUSR8=ADR

Il valore dell'espressione intera è l'indirizzo di inizio della routine USR designata. Quando il sottoprogramma USR viene chiamato, il registro accumulatore (A) contiene un valore che specifica il tipo di argomento (intero, precisione singola e doppia, stringa) assegnato alla funzione USR. Questo valore dà anche la lunghezza del descrittore per quel tipo di argomento:

Valore in A	Significato
2	numero intero di due bytes rappresentato in complemento a due con segno.
3	stringa
4	numero di quattro bytes in virgola mobile in precisione singola.
8	otto bytes in precisione doppia.

Esaminiamo ora alcuni esempi per i vari tipi di argomento. In questi esempi adotteremo la seguente convenzione:

XX indica il contenuto della coppia di registri XX, dove XX può essere una qualsiasi coppia di registri Z80, ad esempio HL.

(XX) indica il contenuto della locazione di memoria specificata dal valore di XX.

Intero PRINT USRØ(I%)

A 2

HL puntatore al valore di I%, che è memorizzato nella rappresentazione in complemento a due con segno

(HL) byte meno significativo di I%

(HL+1) byte più significativo

Singola precisione PRINT USRØ(4.7)

A 4

HL puntatore al valore dell'argomento (in questo caso 4.7)

(HL) gli 8 bits meno significativi della mantissa

(HL+1) gli 8 bits intermedi della mantissa.

(HL+2) i 7 bits più significativi della mantissa ed il segno del numero.

(HL+3) esponente in eccesso a 200 ottale. Se (HL+3)=200, l'esponente è 0. Se (HL+3)=0, il numero é zero.

Doppia precisione PRINT USR0(J#)

A 8

HL puntatore al valore di J#.

(HL-4) gli 8 bits meno significativi della mantissa.
.....

.....

(HL+2) come per la singola precisione.

(HL+3) come per la singola precisione.

Stringa PRINT USR0(A$)

A 3

DE indirizzo del descrittore della stringa A$.

(DE) lunghezza della stringa (0÷255 decimale).

(DE+1) gli 8 bits meno significativi dell'indirizzo in memoria del primo byte della stringa.

(DE+2) gli 8 bits più significativi ecc. ecc..

Normalmente il valore di ritorno da una funzione USR sarà dello stesso tipo (intero, stringa, ecc.) di quello dell'argomento che era stato passato. Tuttavia chiamando la routine MAKINT, il cui indirizzo è memorizzato nella locazione 6, si otterrà come valore della funzione l'intero contenuto in HL, forzando così ad essere intero il valore restituito dalla funzione. Per ritornare dalla funzione occorre eseguire la sequenza sotto riportata:

```
PUSH HL          salva il valore da ritornare
LD   HL,(6)      carica in HL l'indirizzo della
                 MAKINT
EX   (SP),HL     ripristina HL e salva nello stack
                 l'indirizzo della MAKINT
RET              ritorna dalla funzione saltando
                 alla MAKINT.
```

Anche l'argomento di una funzione può essere forzato ad un valore intero, indipendentemente dal suo tipo. Per questo è necessario richiamare il sottoprogramma FRCINT, il cui indirizzo è memorizzato nella locazione 4, per ottenere in HL il valore intero dell'argo-

mento:

```
        LD    HL,SUB1   } carica in HL l'indirizzo della
                        } continuazione del sottoprogram-
        PUSH  HL        } ma e salvalo nello stack

        LD    HL,(4)    carica in HL l'indirizzo di FRCINT

        JP    (HL)      chiama FRCINT

SUB1    .....

        .....

        RET
```

In Appendice E il lettore potrà trovare ulteriori informazioni sulle funzioni USR.

5-4 Il Comando EDIT

Il comando EDIT permette di apportare modifiche ed aggiunte alle linee di programma esistenti senza dover ribattere l'intera linea tutte le volte. I comandi battuti in fase di EDIT non hanno, di regola, eco. Questo significa che, normalmente, non vengono stampati o visualizzati sullo schermo del terminale video quando vengono battuti dall'operatore. Quasi tutti i comandi

possono essere preceduti da un fattore di ripetizione opzionale, che può essere usato per ripetere il comando un certo numero di volte. Questo fattore di ripetizione deve essere compreso fra Ø e 255 (Ø è equivalente a 1) e se è omesso viene assunto uguale ad uno. Negli esempi che seguono una 'n' minuscola rappresenta il fattore numerico di ripetizione. In questo paragrafo, che descrive i comandi disponibili in fase di EDIT, il termine cursore si riferisce ad un puntatore posizionato su un carattere della linea sulla quale si sta lavorando.

Per correggere una linea è necessario battere EDIT seguito dal numero della linea desiderata e dal ritorno carrello. In risposta l'elaboratore stampa il numero della linea richiesta seguito da uno spazio. A questo punto il cursore è posizionato a sinistra del primo carattere della linea.

Se viene battuto un comando non riconosciuto come comando EDIT, il calcolatore stampa un bell ed il comando è ignorato.

Per maggior chiarezza dell'utente negli esem-

pi che seguono alcune linee sono precedute dalla frase "l'elaboratore stampa". In esse è mostrato come appare la linea dopo ogni comando.

a. Come Muovere il Cursore. Battendo uno spazio si sposta il cursore di un posto a destra, provocando contemporaneamente la stampa del carattere oltrepassato dal cursore. Se prima del carattere spazio viene battuto un numero (n'spazio'), il cursore si sposterà a destra di n caratteri stampandoli sul terminale. La pressione del Rubout farà sì che sia stampato il carattere immediatamente precedente la posizione del cursore, mentre questo si sposterà a ritroso.

b. Come Inserire Caratteri.

N.B. L'inserzione dei caratteri è terminata battendo il tasto Escape (o Altmode su certi terminali). Si deve prestare attenzione al fatto che il comando EDIT, quando siamo in modo Insert, non è interrotto dal Control/C, che anzi viene introdotto nella linea sotto correzio-

ne. Per tale motivo Control/C non dovrebbe essere usato nel comando EDIT.

I Inserisce nuovi caratteri nella linea in fase di correzione. Ogni carattere battuto dopo la I è inserito alla posizione corrente del cursore e stampato sul terminale. Per terminare l'inserzione di caratteri si batte Escape. Se cerchiamo di inserire un carattere che renderebbe la linea più lunga di 255 caratteri, viene inviato al terminale un Control/G (bell) ed il carattere non è stampato.
Una freccia sinistra (o un Rubout) battuta durante il comando Insert elimina il carattere a sinistra del cursore. Con questo procedimento è possibile cancellare tutti i caratteri fino all'inizio della linea, e per ogni carattere eliminato verrà stampata sul terminale una freccia sinistra. Se non ci sono più caratteri alla sinistra del cursore, invece di una freccia sinistra viene stampato un bell. Se durante il comando Insert inviamo da tastiera un ritorno car-

rello, otteniamo lo stesso risultato che avremmo ottenuto battendo un Escape seguito dal ritorno carrello. Ossia vengono stampati tutti i caratteri a destra del cursore e la linea di cui avevamo richiesto l'EDIT è rimpiazzata dalla linea corretta.

X Il comando X è molto simile al comando I, eccetto che tutti i caratteri a destra del cursore vengono stampati ed il cursore stesso si posiziona al termine della linea. A questo punto si entra automaticamente nel modo Insert. L'utilità del comando X risulta evidente quando si pensa all'eventualità di dover aggiungere nuove frasi al termine della linea esistente. Vediamo con un esempio l'uso dettagliato di tale comando:

```
L'operatore batte       EDIT 50 (ritorno carrello)
L'elaboratore stampa    50
L'operatore batte          X
L'elaboratore stampa    50 A=A+1
```

L'operatore batte :Y=Y+1 (r. c.)

L'elaboratore stampa 5Ø A=A+1:Y=Y+1

Nell'esempio precedente la linea 5Ø era in origine:

```
5Ø A=A+1
```

che è stata sostituita dalla:

```
5Ø A=A+1:Y=Y+1
```

H H è come X, tranne che tutti i caratteri a destra del cursore vengono eliminati (e non sono stampati). A questo momento è introdotto automaticamente il modo Insert. Il comando H è soprattutto utile quando le ultime frasi di una linea devono essere sostituite con altre nuove.

c. Come Eliminare Caratteri.

D Per cancellare n caratteri a destra del cursore è sufficiente battere nD. Se n é omesso, viene assunto pari ad 1. Se a destra del cursore ci sono meno di n caratteri, sono eliminati solo i caratteri della linea senza provocare danni

al resto del programma. Il cursore è posizionato a destra dell'ultimo carattere eliminato. I caratteri cancellati vengono visualizzati racchiusi fra barre contrarie. Anche per questo comando vediamo un esempio particolareggiato:

```
L'operatore batte       20 X=X+1:REM INCREMENTA X
L'operatore batte       EDIT 20 (r. c.)
L'elaboratore stampa    20
L'operatore batte          6D (r. c.)
L'elaboratore stampa    20 \X=X+1:\REM INCREMENTA X
```

La nuova linea 20 non contiene più i caratteri racchiusi fra le barre contrarie.

d. Ricerca d Caratteri.

S Il comando nSy ricerca all'interno della linea, di cui è stato richiesto l'EDIT, la presenza n-ma del carattere y. Al solito se n non è specificato, viene assunto uguale a 1. La ricerca salta il primo carattere a destra del cursore e inizia dal secondo. Tutti i caratteri che sono

via via oltrepassati durante la ricerca sono stampati sul terminale. Se la ricerca dà esito positivo, il cursore si fermerà alla destra del carattere specificato e tutti i caratteri alla sua sinistra saranno stati stampati. Altrimenti il cursore si posizionerà alla fine della linea. Esempio:

```
L'operatore batte      50 REM INCREMENTA X
L'operatore batte      EDIT 50
L'elaboratore stampa   50
L'operatore batte         2SE
L'elaboratore stampa   50 REM INCR
```

K nKy equivale ad S tranne che tutti i caratteri oltrepassati in fase di ricerca vengono eliminati. I caratteri cancellati sono racchiusi fra barre contrarie. Esempio:

```
L'operatore batte      50 X=X+1:Y=Y+1
L'operatore batte      EDIT 50
L'elaboratore stampa   50
L'operatore batte         KY
L'elaboratore stampa   50 \X=X+1:\
```

e. Sostituzione di Testi.

C Un carattere di una linea può essere sostituito tramite il comando Cy, che cambia il carattere a destra del cursore nel carattere y. Sul terminale viene stampato y ed il cursore avanza di una posizione. Per sostituire n caratteri di una linea con dei nuovi caratteri battuti da terminale, è possibile usare il comando nCy (vedi l'esempio seguente). Se si tenta di cambiare un carattere che non esiste, si esce automaticamente dal modo Change. Esempio:

```
L'operatore batte      10 FOR I=1 TO 100
L'operatore batte      EDIT 10
L'elaboratore stampa   10
L'operatore batte         2S1
L'elaboratore stampa   10 FOR I=1 TO
L'operatore batte                    3C256
L'elaboratore stampa   10 FOR I=1 TO 256
```

f. Uscire e Ripartire.

Ritorno Carrello Termina la fase di editing e stampa il resto della linea. La linea corretta rimpiazza quella originale.

E Ha un effetto analogo al ritorno carrello tranne che non viene stampato il resto della linea.

Q Ripristina la linea originale e provoca il ritorno del BASIC al livello comandi. I cambiamenti non hanno effetto fintanto che non è battuto unritorno carrello o una E? cosicchè il comando Q permette all'utente di ripristinare la linea originale senza alcun cambiamento.

L L causa la stampa del resto della linea, quindi viene stampato il numero di linea e riparte l'editing dall'inizio di tale linea. Il cursore sarà posizionato a sinistra del primo carattere. Il comando L permette perciò di visualizzare lo

effetto dei cambiamenti portati ad una linea. Esempio:

```
L'operatore batte      50 REM INCREMENTA X
L'operatore batte      EDIT 50
L'elaboratore stampa   50
L'operatore batte         2SM
L'elaboratore stampa   50 REM INCRE
L'operatore batte                  L
L'elaboratore stampa   50 REM INCREMENTA X
                       50
```

A Il comando A fa sì che la linea originale sia ripristinata e l'editing riparta dall'inizio della linea stessa. Ad esempio:

```
L'operatore batte      20 AB=25
L'operatore batte      EDIT 20
L'elaboratore stampa   20
L'operatore batte         5D
L'elaboratore stampa   20 \AB=25\
L'operatore batte                  A
```

5-30

L'elaboratore stampa 20 \AB=25\

20

Nell'esempio precedente l'utente aveva commesso un errore cancellando AB=25. Ad esempio egli voleva battere 1D invece di 5D. Come risultato del comando A è reimmessa la linea 10 originale pronta per le nuove correzioni.

Con A termina la descrizione dei sottocomandi del comando EDIT. Restano da fare solo alcune considerazioni che riportiamo a conclusione del paragrafo.

Ogni volta che, durante l'esecuzione di un programma, è rilevato un errore di sintassi (SYNTAX ERROR) il BASIC si porta immediatamente in modo EDIT per la linea nella quale si è verificato l'errore, come se fosse stato battuto il comando EDIT. Esempio:

```
10 PIPPO
RUN
SYNTAX ERROR IN 10
10
```

Al termine dell'editing la linea corretta viene reinserita al posto della vecchia. Tale reinserzione provoca la distruzione di tutti i valori delle variabili. Per preservare questi valori per eventuali esami, è possibile uscire dal modo EDIT tramite il comando Q, dopo che è stato stampato il numero di linea. In tal modo il BASIC tornerà al livello comandi ed i valori delle variabili saranno salvati.

Le facilitazioni offerte dal comando EDIT possono essere usate sulla linea in corso di battitura. Per ottenere questo è sufficiente battere Control/A al posto del ritorno carrello. L'elaboratore risponderà con un ritorno carrello, un punto esclamativo e uno spazio. A questo punto il cursore è posizionato sul primo carattere della linea, ed è possibile usare tutti i sottocomandi dell'EDIT con l'eccezione del Control/A. Esempio:

```
L'operatore batte       10 IF X GOTO""/A
L'elaboratore stampa    !
L'operatore batte         S"          2C22
```

L'elaboratore stampa ! 10 IF X GOTO 22

Il numero di linea corrente può essere specificato da un punto (.) in qualsiasi comando che richieda un numero di linea. Esempio:

L'operatore batte 10 FOR I=1 TO 10

L'operatore batte EDIT .

L'elaboratore stampa 10

5-5 La frase PRINT USING

L'istruzione PRINT USING permette di ottenere un particolare formato di stampa, cioè dati numerici ben incolonnati o spaziature diverse da quelle usuali. Il formato della frase è il seguente:

PRINT USING 'stringa';'lista di valori'

La stringa è un'espressione alfanumerica che rappresenta l'immagine secondo cui deve essere stampata la linea. La lista contiene le costanti, i nomi di variabili e le espressioni da stampare separate una dall'al-

tra da punteggiatura come nella frase PRINT.

La stringa immagine suddivide la riga di stampa in campi che permettono di visualizzare numeri in forma intera o decimale o esponenziale, oppure stringhe oppure costanti dirette.
Distinguiamo i seguenti campi:

a. Campi Stringhe.

"!" Specifica che il campo stringa è di un carattere. E' destinato a contenere il carattere più a sinistra della stringa corrispondente.

"\n spazi\" Specifica che il campo stringa è di 2+n caratteri. E' destinato a contenere stringhe di lunghezza maggiore o uguale a due. Le stringhe in uscita occupano il campo stringa da sinistra verso destra con eventuale troncamento della parte eccedente a destra.

Il tentativo di stampare un numero in un campo stringa provoca l'errore TYPE MISMATCH.

Esempi:

```
10 A$="ABCDE":B$="FGH"
20 PRINT USING "!";A$;B$
30 PRINT USING "\  \";B$;A$
```

La stampa risultante sarà:

```
AF
FGH ABCD
```

Si noti che, quando si usa "!", è stampata solo la prima lettera di ogni stringa. Se le barre racchiudono due spazi, vengono stampate, come nel nostro esempio, quattro lettere di ogni stringa (è stampato uno spazio in più per B$ perchè ha solo tre lettere). Le lettere eccedenti sono invece ignorate.

b. Campi Numerici. Tramite l'uso della frase PRINT USING, le stampe di dati numerici possono essere rese conformi alle più svariate esigenze. Le stringhe per il formattamento dei campi numerici sono costruite a partire dai seguenti caratteri:

I campi numerici interi sono specificati da segni # , ognuno dei quali rappresenta una cifra. I numeri stampati sono allineati a destra; se il numero è più piccolo del previsto, saranno stampati anteriormente tanti spazi quanti sono necessari per riempire l'intero campo.

Il punto decimale può essere specificato in ogni posizione del campo. I campi numerici decimali vengono definiti per contenere numeri con parte decimale; la loro stringa immagine è costituita da tanti segni # quante sono le cifre che costituiscono la parte intera del numero, dal punto decimale e da un insieme di segni # pari al numero di cifre decimali che si vogliono visualizzare. I numeri stampati con una maschera di questo tipo vengono incolonnati sul punto decimale del campo. Se la parte decimale è più corta di quanto previsto nella stringa immagine, vengono aggiunti i necessari zero significativi; se invece è più lunga si ha un troncamento con ar-

rotondamento per eccesso se la prima cifra eliminata è maggiore o uguale a cinque, per difetto in caso contrario.

Il seguente programma illustra le regole esposte:

```
10 INPUT A$,A
20 PRINT USING A$;A
30 GOTO 10
RUN
? ##,12
 12
? ###,12
  12
? #####,12
    12
? ##.##,12
 12.00
? ###.,12
  12.
? #.###,.02
 0.020
```

```
? ##.#,2.36
  2.4
? ###,-12
 -12
? #.##,-.12
 -.12
? ####,-12
  -12
```

+ Il segno + può essere visualizzato all'inizio o alla fine del campo numerico. Se il numero è positivo, il segno + sarà stampato al posto specificato; se il numero è negativo, sarà stampato il segno -.

- Il segno -, se usato alla destra del campo numerico, causa la stampa del segno - alla destra del numero se è negativo. Se il numero è positivo, sarà stampato uno spazio.

** Il segno ** posto all'inizio di un campo numerico causa la stampa degli asterischi necessa-

ri a riempire gli spazi del campo previsto non utilizzati dal numero. Inoltre gli ** ampliano il campo di due cifre.

$$ Il segno $$ posto all'inizio di un campo numerico provoca la stampa del segno $ nello spazio che precede il numero da stampare. $$, inoltre, amplia il campo di due cifre, ma il segno $ occupa uno di questi due spazi.

$ Il segno **$ posto all'inizio di un campo numerico fa agire i due segni visti sopra (e $$) sul numero da stampare. Si applicano tutte le regole viste in precedenza, eccetto che il **$ amplia il campo di tre cifre una delle quali è occupata dal segno $.

Una virgola posta alla sinistra del punto decimale, in un campo numerico, causa la stampa di una virgola, ogni tre cifre, alla sinistra del punto decimale del numero da stampare. La virgola inoltre amplia il campo di una cifra.

Una virgola posta alla destra del punto decimale, in un campo numerico, è considerata come parte della stringa ed è trattata come carattere da stampare.

^^^^ Formato esponenziale. Campi di questo tipo vengono utilizzati per visualizzare numeri in forma esponenziale. La loro immagine é quella di un campo numerico decimale immediatamente seguito da ^^^^ che hanno lo scopo di riservare lo spazio per la parte esponenziale della notazione (E+XX). La posizione del punto decimale è quella indicata nella stringa immagine, e la rappresentazione del numero è tale che il secondo segno ≠ è occupato dalla cifra più significativa del numero. A meno che non sia usato un + a inizio numero oppure un + o un - in coda, il primo segno ≠ è riservato per stampare uno spazio o un segno -. Se il numero é più grande di quanto previsto nel campo immagine, si ha un arrotondamento per difetto o per

eccesso come nei campi decimali. Esempi:

```
PRINT USING "(##^^^^)";13,17,-8
( 1E+01)( 2E+01)(-8E+00)
*
PRINT USING"(.######^^^^-)";12345,-123456
(.123450E+05 )(.123456E+06-)
*
PRINT USING "(+.##^^^^)";123,-126
(+.12E+03)(-.13E+03)
```

% Se il numero da stampare è più grande di quanto previsto nel campo immagine, si ha la stampa del carattere % seguito dal numero nel formato standard del Child BASIC. Se l'arrotondamento di un numero porta a superare il campo previsto, si ha ha la stampa di % seguito dal numero arrotondato. Se, ad esempio, A=.999, allora

```
PRINT USING ".##",A
```

provocherà la stampa

```
%1.00
```

Se il numero di cifre previste supera 24, si ha l'errore ILLEGAL FUNCTION CALL.

Il seguente programma illustra le regole precedenti.

```
Programma:   10 INPUT A$,A
             20 PRINT USING A$,A
             30 GOTO 10
             RUN
```

Il computer all'inizio stamperà un ?. Inviate il campo numerico previsto e la lista. L'uscita sarà visualizzata come segue:

```
? +≠,9
+9
? +≠,10
%+10
? ≠≠,-2
-2
? +≠,-2
-2
```

```
? ≠,-2
%-2
? +.≠≠≠,.Ø2
+Ø2Ø
? ≠≠≠≠.≠,1ØØ
 1ØØ.Ø
? ≠≠+,2
 2+
? QUESTO E' IL NUMERO ≠≠,2
QUESTO E' IL NUMERO  2
? ≠≠≠≠,44444
%44444
? **≠≠,1
***1
? **≠≠,12
**12
? **≠≠,123
*123
? **≠≠,1234
1234
```

```
? **##,12345
%12345
? **,1
*1
? **,22
22
? **.##,12
12.00
? **#####,1
*****1
```

(nota:$ fisso)

```
? $####.##,12.34
$  12.34
```

(nota:$ mobile)

```
? $$####.##,12.56
 $12.56
? $$.##,1.23
$1.23
? $$.##,12.34
%$12.34
? $$###,0.23
   $0
```

```
? $$###.##,0
     $0.00
? **$###.##,1.23
****$1.23
? **$.##,1.23
*$1.23
? **$###,1
****$1
```

```
? #,6.9
 7
? #.#,6.99
 7.0
? ##-,2
 2
? ##-,-2
 2-
? ##+,2
 2+
? ##+,-2
 2-
```

```
? ##^^^^,2
 2E+00
? ##^^^^,12
 1E+01
? #####.###^^^^,2.45678
 2456.780E-03
? #.###^^^^,123
 0.123E+03
? #.##^^^^,-123
-.12E+03
? "#####,###.#",1234567.89
 1,234,570.0
```

Premendo Control/C il programma si arresterà.

VI - LISTE ED ELENCHI

6-1 Comandi

I comandi dirigono il Child BASIC nella gestione della memoria e dell'ingresso/uscita, nella lista e correzione di programmi, ed in generale offrono varie facilitazioni per supportare l'esecuzione dei programmi. Il Child BASIC accetta comandi dopo aver stampato * per indicare appunto che si trova al livello comandi. Nel seguito sono riportati i comandi in ordine alfabetico. Per ogni comando sono indicati il formato e le principali caratteristiche.

AUTO°'NN'°°,°'MM'°° Provvede all'inserzione automatica dei numeri di linea. NN è il numero della prima linea ed MM l'incremento fra linee successive.

CLEAR Pone a zero tutte le variabili del programma.

CLEAR°'espressione'° Ha lo stesso effetto di

CLEAR, ma pone lo spazio riservato alle variabili alfanumeriche pari al valore dell'espressione. Se l'argomento non è specificato, tale spazio rimane inalterato. Al momento del caricamento del BASIC lo spazio è fissato in 200 bytes.

CLOAD'espressione alfanumerica' Provoca il caricamento in memoria del programma, registrato su nastro magnetico, designato dal primo carattere dell'espressione alfanumerica. Esempio: CLOAD"A". Prima del caricamento del programma viene inviato un comando NEW.

CLOAD?'espressione alfanumerica' Confronta il programma in memoria con il file in cassetta avente lo stesso nome. Se sono uguali, il BASIC stampa OK, altrimenti stampa NO GOOD.

CLOAD*'nome di matrice' Carica da nastro magnetico la matrice specificata. Questo comando può anche essere usato come frase di programma.

CONT Continua l'esecuzione di un programma dopo che è stato battuto un Control/C, o è stata eseguita una frase STOP o END. L'esecuzione riprende dalla frase successiva a quella in cui si era avuto l'arresto, a meno che non fosse stato interrotto un ingresso da terminale. In tal caso l'esecuzione riprende con la ristampa dell'avviso (? o stringa d'avviso). Il comando CONT è utile soprattutto in fase di debug, specialmente quando c'è il sospetto di essere entrati in un loop infinito. Un loop infinito è una serie di frasi che vengono ripetute indefinitamente, senza quindi poterne uscire. Battendo Control/C si interrompe l'esecuzione ed il BASIC si porta al livello comandi. A questo punto è possibile usare varie frasi in modo diretto per stampare risultati intermedi, cambiare il valore delle variabili, e così via. Per ripartire si adopera il comando CONT od anche la frase GOTO usata in modo diretto, così da riprendere l'esecuzione dal numero di linea specificato. Si noti che l'esecuzione non può continuare se il programma è stato modificato durante l'interruzione.

CSAVE'espressione alfanumerica' Fa sì che il programma attualmente in memoria sia salvato su nastro magnetico sotto il nome specificato dal primo carattere dell'espressione alfanumerica. Un programma da caricare su nastro deve sempre terminare con la frase END.

CSAVE*'nome di matrice' Salva su nastro la matrice specificata. Può essere usato come frase di programma.

DELETE'numero di linea' Elimina la linea di programma avente il numero specificato. Se tale linea non esiste, si ha l'errore ILLEGAL FUNCTION CALL.

DELETE-'numero di linea' Cancella ogni linea del programma corrente fino a quella specificata inclusa. Se tale linea non esiste, viene segnalato l'errore ILLEGAL FUNCTION CALL.

DELETE'numero di linea'-'numero di linea' Elimina tutte le linee di programma dal primo numero al

secondo compreso. Al solito se nessuna linea ha il secondo numero, si ha un ILLEGAL FUNCTION CALL.

EDIT'numero di linea' Permette di correggere la linea indicata senza influenzare le altre linee. Il comando EDIT gode di un esteso e potente insieme di sottocomandi discussi in dettaglio nel par. 5-4.

LIST Lista il programma attualmente in memoria a partire dalla linea con il numero minore. La lista è terminata dalla fine del programma o dalla pressione di Control/C.

LIST°'numero di linea'° Stampa, se esiste, la linea specificata.

LIST°'numero di linea'°°-'numero di linea'° Consente diverse possibilità di lista:

1. Se il secondo numero è omesso, lista tutte le linee di numero maggiore o uguale a quello specificato.

2. Se è omesso il primo numero, lista tutte le linee dall'inizio del programma fino alla linea specifi-

cata inclusa.

3. Se sono usati entrambi i numeri, lista tutte le linee dal primo numero al secondo incluso.

LLIST°'n. di linea'°°-'n. di linea'° Offre le stesse opzioni del precedente, ma la lista viene effettuata sulla stampante.

NEW Cancella il programma corrente e mette a zero tutte le variabili. E' usato prima di battere un nuovo programma.

RENUM°'NN'°,'MM'°,'II'°°° Consente di dare una nuova numerazione alle linee di un programma, senza però alterarne l'ordine. NN è il numero che viene attribuito alla prima delle linee rinumerate. Le linee minori di MM non vengono rinumerate. II è l'incremento che viene dato fra linea e linea nella nuova sequenza.

RUN°'numero di linea'° Inizia l'esecuzione

del programma attualmente in memoria a partire dalla linea specificata. Se il numero di linea è omesso, l'esecuzione comincia da quello più basso.

6-2 Frasi

Nel paragrafo che segue è riportato un elenco in ordine alfabetico delle frasi BASIC. Nell'elenco X e Y stanno per una qualunque espressione valida per il Child BASIC, mentre I e J stanno per espressioni i cui valori sono troncati ad un intero. Infine V e W rappresentano nomi di variabili.

Nome Formato

DATA DATA'lista'

Specifica i dati che devono essere letti da una frase READ. Gli elementi della lista possono essere numeri o stringhe e devono essere separati da virgole.

DEF DEF FNV('W')='X'

Definisce una funzione utente. Il nome della funzione è FN seguito da un nome di variabile legale. La fun-

zione definita può essere anche di tipo alfanumerico. Le definizioni sono ristrette ad una linea (255 caratteri).

DEFDBL DEFDBL r

Definisce variabili in doppia precisione. 'r' è una lettera o un insieme di lettere. Ade esempio DEFDBL D significa che i nomi di variabile che cominciano per D devono essere del tipo doppia precisione. DEFDBL I-N significa che i nomi di variabile che iniziano con le lettere da I a N sono in doppia precisione.

DEFINT DEFINT r

Come sopra, ma definisce variabili intere.

DEFSNG DEFSNG r

Come sopra, ma definisce variabili in precisione singola.

DEFSTR DEFSTR r

Come sopra, ma definisce variabili alfanumeriche.

DEFUSR DEFUSR°'cifra'°='X'

Definisce l'indirizzo di partenza di un sottoprogramma in linguaggio macchina. Sono permessi fino a dieci sottoprogrammi.

DIM DIM'V'('I'°,J...°)°,...°

Assegna la memoria alle variabili con indice (matrici). Con una frase DIM possono essere dimensionate più matrici con l'avvertenza di non superare la lunghezza di una linea. Il valore di ogni espressione dà il massimo indice possibile. L'indice più piccolo è Ø. Senza una frase DIM si assume che una matrice abbia un indice massimo di 1Ø per ogni dimensione a cui si fa riferimento. Ad esempio A(I,J) è considerata composta di 121 elementi, da A(Ø,Ø) ad A(1Ø,1Ø) a meno che non sia dimensionata altrimenti da una frase DIM.

END END

Termina l'esecuzione di un programma.

ERASE ERASE'V'°,'W'...°

Elimina le matrici specificate. Le matrici possono es-

sere ridimensionate oppure lo spazio può essere reso disponibile per altri usi.

FOR FOR'V'='X'TO'Y'(STEP'Z')

Permette l'esecuzione ripetitiva delle stesse frasi. La prima esecuzione pone V=X, dopodichè si procede normalmente, finchè non si incontra una frase NEXT. A questo punto Z viene sommato a V, poi se Z<Ø e V>=Y, o se Z>Ø e V<=Y, il BASIC salta indietro alla frase dopo FOR. Altrimenti l'esecuzione continua con la frase dopo NEXT.

GOTO GOTO'nnnnn'

Salto incondizionato al numero di linea specificato.

GOSUB GOSUB'nnnnn'

Salto incondizionato al sottoprogramma che inizia alla linea nnnnn.

IF...GOTO IF'X'GOTO'nnnnn'

Come IF...THEN (v. dopo) eccetto che GOTO può essere seguito solo da un numero di linea e non da un'altra

frase.

IF...THEN°ELSE° IF'X'THEN'Y'°ELSE'Z'°

o IF'X'THEN'frase'°:frase...°

°ELSE'frase'°:frase...°°

Se il valore di X è diverso da zero, salta alla linea o alla frase (i) dopo THEN. Altrimenti salta alla linea o alla frase (i) dopo ELSE. Se ELSE è omesso, e il valore di X è zero, l'esecuzione procede dalla linea dopo IF... THEN.

INPUT INPUT'V'°,'W'...°

Provoca una richiesta di ingresso da terminale. I valori battuti sono assegnati alle variabili della lista.

LET LET'V'='X'

Assegna alla variabile V il valore dell'espressione X. La parola LET è opzionale.

LINE INPUT LINE INPUT "stringa d'avviso";

'nome di variabile alfanumerica'

Stampa la stringa d'avviso sul terminale ed assegna

tutto l'ingresso, dalla fine dell'avviso fino al ritorno carrello, alla variabile alfanumerica specificata. Nessun altro avviso è stampato, se è omessa la stringa d'avviso.

LPRINT LPRINT X°,Y...°

Come PRINT, ma esce sulla stampante. I caratteri 'line feed' (interlinea) all'interno delle stringhe sono ignorati. Dopo l'ottantesimo carattere di una linea è stampato automaticamente un ritorno carrello.

LPRINT USING LPRINT USING'stringa';'lista'

Come PRINT USING, ma esce sulla stampante. per una descrizione dettagliata, vedi par. 5-5.

MID$ MID$('X$','I'°,'J'°)=Y$

Parte della stringa X$ è sostituita da Y$. La sostituzione inizia con l'i-mo carattere di X$ e continua finchè Y$ è esaurito, o è raggiunta la fine di X$ o infine sono stati rimpiazzati J caratteri, a seconda dell'evento che si verifica prima. Se I è maggiore della lunghezza della stringa X$, viene segnalato lo

errore ILLEGAL FUNCTION CALL.

NEXT NEXT°'V','W'...°

E' l'ultima frase di un ciclo FOR. V è la variabile del ciclo più recente, W del penultimo e così via. Una frase NEXT senza variabile termina il ciclo più recente.

ON ERROR GOTO ON ERROR GOTO'numero di linea'

Quando si ha un errore, salta alla linea specificata. Pone la variabile ERR al codice di errore e la variabile ERL al numero di linea in cui si è avuto l'errore. Per un elenco dei codici di errore vedi par. 6-5. ON ERROR GOTO Ø (o senza numero) disabilita la rivelazione di errore.

ON...GOTO ON'I'GOTO'lista di numeri di linea'

Salta alla linea il cui numero occupa la I-ma posizione nella lista. Gli elementi della lista sono separati da virgole. SeI=Ø o maggiore del numero di elementi della lista, l'esecuzione continua alla fra-

se successiva. Se I non è compreso fra Ø e 255, si ha un errore.

ON...GOSUB ON'I'GOSUB'lista'

Come ON...GOTO eccetto che gli elementi della lista sono i numeri delle linee iniziali di sottoprogrammi.

OUT OUT'I','J'

Emette il byte J sul port I. Dev'essere Ø<=I<=255 e Ø<= J<=255.

POKE POKE'I','J'

Memorizza il byte J nelle locazione di memoria derivata da I. I limiti sono Ø<=J<=255 e -32768<I<65535. Se I è negativo, l'indirizzo è 65536+I; se I è positivo, l'indirizzo è I.

PRINT PRINT'X'°,'Y'...°

Provoca la stampa sul terminale dei valori delle espressioni della lista. La spaziatura è determinata dalla punteggiatura nel modo ora descritto. La linea è considerata suddivisa in zone di 14 colonne. Se le

espressioni sono separate da virgole, la stampa successiva comincia dall'inizio della successiva zona di 14 colonne; se l'elemento separatore è un punto e virgola, la stampa inizia immediatamente dopo la precedente. In tutti gli altri casi la stampa riprende dall'inizio della riga successiva.

PRINT USING PRINT USING'stringa';'lista'

Stampa i valori delle espressioni della lista in un formato dipendente dalla stringa. La stringa è un'espressione che rappresenta la linea da stampare. La lista contiene le costanti, i nomi di variabile e le espressioni da stampare. Gli elementi della lista sono separati da punteggiatura come nella frase PRINT. Per un elenco dei caratteri della stringa e delle loro funzioni, vedi il par. 5-5.

READ READ'V'°,'W'...°

Assegna alle variabili i valori elencati nella frase DATA. I valori sono assegnati in sequenza a partire dal primo valore della prima frase DATA.

REM REM°'commento'°

Permette di inserire commenti nel programma. Non viene eseguita, ma un'altra frase può saltare ad essa. I commenti possono anche essere aggiunti alla fine di una linea, separati dal resto tramite un apice (').

RESTORE RESTORE

Permette di rileggere i dati delle frasi DATA. La prima frase READ dopo un RESTORE comincia a leggere il primo dato della prima frase DATA.

RESUME RESUME°'numero'°

Riprende l'esecuzione del programma dalla linea specificata, dopo la routine di trattamento dell'errore. Se il numero è omesso o è zero, riprende dalla frase dove si è avuto l'errore. RESUME NEXT provoca la ripresa del programma dalla frase successiva a quella dove è avvenuto l'errore.

RETURN RETURN

Termina un sottoprogramma, saltando alla frase che se-

gue il più recente GOSUB.

STOP STOP

Arresta l'esecuzione del programma. Il BASIC passa al livello comandi e stampa BREAK IN LINE nnnnn.

SWAP SWAP'V','W'

Scambia i valori delle variabili specificate. Ovviamente le variabili devono essere dello stesso tipo.

TROFF TROFF

Esce dal modo debug, in cui era entrato tramite la frase TRON, per rientrare nel modo di esecuzione normale. Lo stesso risultato si ottiene per mezzo del comando NEW.

TRON TRON

Entra nel modo di esecuzione debug. Il BASIC stampa fra parentesi quadre il numero di ogni linea che esegue.

6-3 Funzioni Intrinseche

Il Child BASIC mette a disposizione del program-

matore diverse funzioni algebriche ed alfanumeriche di uso comune, che possono essere chiamate da programma senza ulteriore definizione. Alcune di queste funzioni, se non sono richieste da un programma, possono essere eliminate al momento in cui il BASIC è caricato, per aumentare la memoria a disposizione. Come già per i comandi e per le frasi riportiamo un elenco alfabetico delle funzioni. Come al solito X e Y stanno per espressioni, I e J per espressioni intere ed infine X$ e Y$ per espressioni alfanumeriche.

Funzione	Formato di chiamata

ABS ABS(X)

Dà il valore assoluto dell'espressione X. ABS(X)=X se X>=Ø, -X se X<Ø.

ASC ASC(X$)

Fornisce il codice ASCII del primo carattere della stringa X$.

ATN ATN(X)

Fornisce l'arcotangente di X. Il risultato è in radianti nel campo $-\pi/2 \div +\pi/2$.

CDBL CDBL(X)

Converte X in doppia precisione.

CINT CINT(X)

Converte X ad un intero. Se l'argomento è nel campo -32768÷32767, CINT(X)=INT(X). Altrimenti CINT(X) provocherà l'errore OVERFLOW.

CSNG CSNG(X)

Converte X a singola precisione.

CHR$ CHR$(I)

Ritorna una stringa il cui unico elemento ha il codice ASCII I.

COS COS(X)

Dà il coseno di X. X è in radianti.

ERL ERL

Dà il numero della linea nella quale si è verificato l'ultimo errore.

ERR ERR

Fornisce il codice dell'ultimo errore.

EXP EXP(X)

Ritorna "e" elevato ad X. Deve essere X<=87.3365.

FIX. FIX(X)

Ritorna la parte intera di X. FIX(X) è equivalente a SGN(X)*INT(ABS(X)). La maggiore differenza fra FIX e INT è che, per X negativo, FIX non fornisce il più prossimo intero minore od uguale ad X.

FRE FRE(Ø)

Dà il numero di bytes di memoria non usati dal BASIC. Se l'argomento è una stringa, dà il numero di bytes liberi nello spazio riservato alle variabili alfanumeriche.

HES$ HEX$(X)

Ritorna una stringa che dà la rappresentazione esadecimale dell'argomento decimale.

INP INP(I)

Legge un byte dal port I.

INSTR INSTR(°I,°X$,Y$)

Cerca la prima volta in cui si trova la stringa Y$ all'interno della stringa X$ e ne ritorna la posizione. Un offset opzionale I dà la posizione da cui cominciare la ricerca. Se I è maggiore della lunghezza di X$, se X$ è la stringa nulla o, infine, se Y$ non può essere trovato, INSTR dà come risultato Ø. Se Y$ è nulla, INSTR ritorna I o1.

INT INT(X)

Fornisce il maggiore intero minore o uguale ad X.

LEFT$ LEFT$(X$,I)

Dà l'I-mo carattere, a partire da sinistra, della stringa X$.

LEN LEN(X$)

Dà la lunghezza della stringa X$. Sono contati anche i caratteri non stampabili e gli spazi.

LOG LOG(X)

Calcola il logaritmo naturale di X, che dev'essere ovviamente maggiore di zero.

LPOS LPOS(X)

Dà la posizione corrente della testina della stampante all'interno del buffer della stampante. Non coincide necessariamente con la posizione fisica della testina. L'espressione X deve essere data, ma il suo valore è ignorato.

MID$ MID$(X$,I°,J°)

Senza J preleva i caratteri più a destra di X$ a partire dal carattere I-mo. Se I>LEN(X$), MID$ dà la stringa nulla. I deve essere compreso fra Ø e 255. Con tre argomenti il risultato è una stringa di lunghezza J di caratteri prelevati da X$ a partire dall'I-mo. Se J è maggiore del numero di caratteri di X$ a destra dell'I-

mo, MID$ preleva il resto della stringa. Anche per J i limiti sono Ø÷255.

OCT$ OCT$(X)

Dà una stringa che rappresenta il valore ottale dell'argomento decimale.

RND RND(X)

Genera un numero casuale (RaNDom) compreso fra Ø e1. X<Ø inizia una nuova sequenza di numeri casuali. X>Ø dà il numero successivo della sequenza. X=Ø dà nuovamente l'ultimo numero fornito. Le sequenze iniziate con lo stesso numero negativo sono identiche.

PEEK PEEK(I)

Legge un byte dalla locazione di memoria I.

POS POS(I)

Dà la colonna sulla quale è posizionato al momento il cursore del terminale. La posizione più a sinistra è Ø.

RIGHT$ RIGHT$(X$,I)

Preleva gli I caratteri più a destra della stringa X$. Se I=LEN(X$) il risultato è X$.

SGN SGN(X)

Se X>Ø, SGN(X)=1; se X=Ø, SGN(X)=Ø; se X<Ø, SGN(X)=-1. Ad esempio, ON SGN(X)+2 GOTO 1ØØ,2ØØ,3ØØ salta a 1ØØ se X è negativo, 2ØØ se X è zero e 3ØØ se X è positivo.

SIN SIN(X)

Calcola il seno del valore, espresso in radianti, di X.

SPACE$ SPACE$(I)

Dà una stringa di spazi di lunghezza I.

SPC SPC(I)

Stampa sul terminale I spazi, con I compreso fra Ø e 255.

SQR SQR(X)

Calcola la radice quadrata di X, con X>=Ø.

STR$ STR$(X)

Ritorna la rappresentazione come stringa del valore di X.

STRING$ STRING$(I,J)

Dà una stringa di lunghezza I, i cui caratteri hanno tutti codice ASCII J.

TAB TAB(I)

Si posiziona alla colonna I del terminale. La posizione più a sinistra è Ø, quella più a destra 71. Se il carrello (od il cursore) si trova già oltre lo spazio I, TAB(I) non ha effetto. Al solito è Ø<=I<=255. La funzione TAB può essere usata solo nelle frasi PRINT e LPRINT.

TAN TAN(X)

Calcola la tangente di X, con X in radianti.

USR USR(X)

Chiama il sottoprogramma in linguaggio macchina defi-

nito dall'utente con argomento X.

VAL VAL(X$)

Dà il valore numerico della stringa X$. Se il primo carattere di X$ non è +, - o una cifra è VAL(X$)=Ø.

VARPTR VARPTR(V)

Ritorna l'indirizzo della variabile data come argomento. Se durante l'esecuzione del programma alla variabile non è stato assegnato un valore, si verifica l'errore ILLEGAL FUNCTION CALL. Il principale uso della variabile VARPTR è di avere l'indirizzo di una variabile semplice o matriciale in modo di passarlo ad un sottoprogramma in linguaggio macchina. Gli indirizzi delle matrici usualmente sono passati specificando VARPTR(A(Ø)), cosicchè si ottiene l'elemento della matrice di indirizzo più basso.

N.B. A tutte le variabili semplici dovrebbero essere assegnati i valori prima di chiamare la VARPTR per una qualunque matrice. In caso contrario l'allocazione

di una nuova variabile semplice provocherà il cambiamento degli indirizzi di tutte le matrici.

6-4 Caratteri Speciali

Nel Child BASIC diversi caratteri del codice ASCII hanno particolari funzioni speciali nel controllo del cursore, nella correzione e nell'interruzione di programmi. Si ricorda che caratteri come Control/C, Control/S, ecc. si ottengono tenendo premuto il tasto Control e battendo la lettera designata.

Nella seguente tabella sono riportati nella colonna di sinistra il carattere battuto ed in quella di destra l'eco prodotta sul terminale.

Battuto come	Stampato come
@	@

Elimina la linea corrente ed esegue un ritorno carrello.

←

Cancella l'ultimo carattere battuto. Se non c'è nessun carattere, invia un ritorno carrello.

_ _ (sottolineatura)

Come il precedente.

Ritorno carrello

Posiziona la testina o il cursore all'inizio della riga successiva.

Control/C ^C

Interrompe l'esecuzione del programma corrente o del comando di lista. Ha effetto dopo l'esecuzione della frase attuale o la lista della linea attuale. Il BASIC si porta al livello comandi e stampa *. L'esecuzione riprende tramite il comando CONT.

: :

Separa le frasi all'interno di una stessa linea.

Control/O ^O

Sopprime tutte le uscite, finchè è trovata una frase INPUT, o è battuto un altro Control/O, o si verifica un errore o, infine, il BASIC ritorna al livello comandi.

? ?

Equivale alla frase PRINT.

Rubout vedi spiegazione

Cancella il carattere precedente in una linea in fase di stesura. Il primo Rubout stampa una barra contraria (\) e l'ultimo carattere battuto. Ogni successivo Rubout stampa il carattere successivo a sinistra. Battendo un nuovo carattere viene stampato nuovamente \ e il nuovo carattere. A questo punto tutti i caratteri fra le due barre sono stati eliminati.

Control/U ^U

Come @.

Control/S

Provoca il momentaneo arresto dell'esecuzione di un programma, finchè è battuto Control/Q o Control/C.

Control/Q

Riprende l'esecuzione di un programma dopo un Control/S. Control/S e Control/Q non hanno effetto di sorta se nessun programma è in corso di esecuzione.

Control/A

Permette l'uso del comando EDIT sulla linea in corso di battitura (v. par. 5-4). Control/A è battuto al posto del ritorno carrello.

Control/I da 1 a 8 spazi

E' il carattere TAB. Fa sì che la testina o il cursore si porti all'inizio del successivo campo di 8 colonne. I campi cominciano alle colonne 1, 9, 17 e così via. Il carattere TAB è particolarmente utile per il formattamento di linee spezzate da interlinee, come mostra il seguente esempio, in cui il carattere in-

terlinea è indicato con 'lf' (Line Feed).

```
100'tab'FOR I=1 TO 10:'lf'
'tab''tab'FOR J=1 TO 10:'lf'
'tab''tab''tab'A(I,J)=0:'lf'
'tab'NEXT J,I'ritorno carrello'
```

viene listato come

```
100     FOR I=1 TO 10:
                FOR J=1 TO 10:
                        A(I,J)=0:
        NEXT J,I
```

Control/G bell

Fa suonare il campanello del terminale.

LINE FEED

Spezza in varie parti una linea lunga. Il risultato è ancora una linea BASIC.

Denota il numero della linea corrente. Può essere u-

sato ogni volta che deve essere specificato un numero di linea.

[,] [,]

Le parentesi quadre sono equivalenti alle tonde per delimitare gli indici di una matrice.

Lettere minuscole

I caratteri alfabetici minuscoli hanno sempre come eco lettere minuscole, ma i comandi LIST e LLIST e le frasi PRINT e LPRINT effettueranno sempre la conversione da minuscolo a maiuscolo, se i caratteri minuscoli non fanno parte di stringhe o commenti.

6-5 Messaggi di Errore

Dopo un errore il BASIC ritorna al livello comandi e stampa *. I valori delle variabili ed il testo del programma restano inalterati, ma il programma non può essere ripreso dal comando CONT. Questo è tuttavia possibile usando la frase GOTO in modo diretto.

Quando si verifica un errore in una frase diretta non viene stampato, ovviamente,nessun numero di linea. I formati dei messaggi di errore sono i seguenti:

frase diretta	?XX ERROR
frase indiretta	?XX ERROR IN YYYYY

dove XX è il messaggio di errore e YYYYY è il numero della linea in cui l'errore si è verificato. Riportiamo di seguito i codici di errore ed i loro significati:

CODICE	MESSAGGIO	NUMERO
BS	SUBSCRIPT OUT OF RANGE	9

E' stato fatto un tentativo di riferirsi ad un elemento di una matrice che è al di fuori delle dimensioni della matrice stessa. Questo errore si verifica anche quando in un riferimento ad una matrice viene usato un numero sbagliato di dimensioni. Ad esempio:

```
LET A(5,5,1)=B
```

quando A è già stata dimensionata da DIM A(10,10).

DD REDIMENSIONED ARRAY 10

Dopo che una matrice è stata dimensionata, viene trovata una nuova frase di dimensionamento per la stessa matrice.

FC ILLEGAL FUNCTION CALL 5

Il parametro passato ad una funzione matematica od alfanumerica è fuori del campo consentito. Gli errori di tipo FC si verificano in seguito a:

1. Un indice di matrice negativo (LET A(-1)=Ø)
2. Un indice di matrice troppo grande (>32767)
3. LOG con argomento negativo o nullo
4. SQR con argomento negativo
5. A^B con A negativo e B non intero
6. Una chiamata ad una USR prima che sia stato immesso l'indirizzo di un sottoprogramma in linguaggio macchina
7. Chiamate a MID$, LEFT$, RIGHT$, INP, OUT, PEEK, POKE, TAB, SPC, STRING$, SPACE$, INSTR od infine ON...GOTO facendo uso di un argomento improprio.

ID ILLEGAL DIRECT 12

La frase DEF usata in modo diretto è illegale.

NF NEXT WITHOUT FOR 1

La variabile di una frase NEXT non corrisponde a nessuna frase FOR eseguita in precedenza.

OD OUT OF DATA 4

Si cerca di eseguire una frase READ, quando tutte le frasi DATA del programma sono già state lette. Il programma tentava di leggere troppi dati o, nel programma, era stato incluso un numero di dati insufficiente.

OM OUT OF MEMORY 7

Il programma è troppo esteso, contiene troppe variabili, troppi cicli FOR, troppe frasi GOSUB o espressioni troppo complicate. Vedi Appendice C.

OV OVERFLOW 6

Il risultato di un calcolo è troppo grande per essere rappresentato nel formato numerico del Child BASIC. Se

si verifica un underflow (risultato troppo piccolo), come risultato viene dato zero e l'esecuzione procede senza che venga stampato nessun messaggio di errore.

SN SYNTAX ERROR 3

Parentesi omesse in un'espressione, carattere illegale in una linea, punteggiatura errata, ecc.

RG RETURN WITHOUT GOSUB 3

E' incontrata una frase RETURN prima che sia stata eseguita in precedenza una frase GOSUB.

UL UNDEFINED LINE 8

Il riferimento alla linea in un GOTO, GOSUB, IF...THEN ...ELSE o DELETE è ad una linea inesistente del programma.

/0 DIVISION BY ZERO 11

Può verificarsi tanto in una divisione intera quanto in una divisione in virgola mobile. Anche zero elevato ad una potenza negativa provoca l'errore DIVISION BY ZERO.

CN CAN'T CONTINUE 17

Si verifica in seguito al tentativo di continuare un programma quando nessun programma esiste, è avvenuto un errore, o dopo che è stata fatta na modifica del programma.

LS STRING TOO LONG 15

E' stato fatto un tentativo di creare una stringa più lunga di 255 caratteri.

OS OUT OF STRING SPACE 14

Le variabili alfanumeriche superano lo spazio di memoria loro riservato. Per assegnare più spazio di memoria alle stringhe o usare stringhe più piccole o minori variabili alfanumeriche si adoperi il comando CLEAR.

ST STRING FORMULA TOO COMPLEX 16

Un'espressione alfanumerica è troppo lunga o troppo complessa. E' necessario spezzarla in due o più parti.

TM TYPE MISMATCH 13

Il lato sinistro di una frase di assegnazione è una va-

riabile alfanumerica e quello destro una numerica o viceversa; oppure ad una funzione che richiede un argomento numerico ne è stato dato uno alfanumerico o viceversa.

UF UNDEFINED USER FUNCTION 18

Viene fatto riferimento ad una funzione utente che non è mai stata definita.

MISSING OPERAND 20

Durante il calcolo di un'espressione viene trovato un operatore privo del suo operando.

NO RESUME 19

Il BASIC è entrato in una routine di trattamento dell'errore, ma il programma è terminato prima di aver incontrato una frase RESUME.

RESUME WITHOUT ERROR 21

E' incontrata una frase RESUME, ma non si era mai entrati in nessuna routine di trattamento dell'errore.

UNPRINTABLE ERROR 22

Esiste una condizione di errore per la quale non c'è

nessun messaggio disponibile. Si riferisce anche ad un errore che si verifica in fase di lettura da unità a cassette.

LINE BUFFER OVERFLOW 23

C'è stato un tentativo di introdurre un programma o una linea di dati che ha troppi caratteri per essere contenuti nel buffer delle linee. E' necessario accorciare la linea o dividerla in più parti.

6-6 Parole Riservate

Alcune parole sono riservate dal BASIC come nomi di frasi, comandi, operatori, ecc. e non possono così essere usate come nomi di variabili e di funzioni. Qua di seguito sono riportate in ordine alfabetico le parole riservate. Si ricordi che oltre quelle sotto elencate, anche i nomi di funzioni intrinseche sono parole riservate.

AND AUTO CLEAR CONT DATA DEF DEFDBL DEFINT
DEFSNG DEFSTR DELETE DIM EDIT ELSE END
ERASE ERL ERR FN FOR GOSUB GOTO IF IMP
INPUT INSTR LET LINE LIST LLIST LPRINT

6-40

MOD NEW NEXT NOT ON OR OUT POKE PRINT READ REM RENUM RESUME RETURN RUN SPACE$ SPC STOP STRING$ SWAP TAB THEN TO TROFF TRON USR VARPTR WIDTH XOR.

APPENDICE A

IL SET DEI CARATTERI ASCII

Bits b_6 b_5 b_4 / b_3 b_2 b_1 b_0 / ROW / COLUMN	$0\ 0\ 0$	$0\ 0\ 1$	$0\ 1\ 0$	$0\ 1\ 1$	$1\ 0\ 0$	$1\ 0\ 1$	$1\ 1\ 0$	$1\ 1\ 1$
	0	1	2	3	4	5	6	7
0 0 0 0 0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0 2	STX	DC2	"	2	B	R	b	r
0 0 1 1 3	ETX	DC3	#	3	C	S	c	s
0 1 0 0 4	EOT	DC4	$	4	D	T	d	t
0 1 0 1 5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0 6	ACK	SYN	&	6	F	V	f	v
0 1 1 1 7	BEL	ETB	'	7	G	W	g	w
1 0 0 0 8	BS	CAN	(	8	H	X	h	x
1 0 0 1 9	HT	EM	)	9	I	Y	i	y
1 0 1 0 10	LF	SUB	*	:	J	Z	j	z
1 0 1 1 11	VT	ESC	+	;	K	[	k	{
1 1 0 0 12	FF	FS	,	<	L	\	l	|
1 1 0 1 13	CR	GS	-	=	M	]	m	}
1 1 1 0 14	SO	RS	.	>	N	^	n	~
1 1 1 1 15	SI	US	/	?	O	_	o	DEL

caratteri non stampabili

APPENDICE B

COME CARICARE ED INIZIALIZZARE IL BASIC

a. Caricamento del BASIC da nastro.

Di seguito è riportata la procedura di caricamento del BASIC da nastro magnetico per le due versioni di unità centrale attualmente in produzione alla General Processor, e cioé per la versione con pannello di controllo esadecimale RTOC e per la versione Turn-Key senza pannello RTOC.

Versione RTOC

1. Accendere il sistema.
2. Premere il pulsante RESET.
3. Premere in sequenza i tasti FUNCT e LOAD. Come conseguenza di questa manovra il pannello RTOC viene disabilitato e si spengono i displays posti su di esso. Si accendono invece i leds "Read" e "Drive 1" dell'interfaccia Hits.
4. Porre il registratore in "PLAY".
5. Al termine della lettura il pannello RTOC vie-

ne riabilitato e si spengono i leds dell'Hits. Sul terminale video appare la scritta "MEMORY SIZE?".

6. In caso di errore di lettura si spengono i leds Hits e sul pannello compare la segnalazione di errore "EEEE C5". E' allora necessario ripartire dal punto 2, dopo avere ovviamente riavvolto il nastro.

Versione Turn-Key

1. Accendere il sistema.
2. Premere il pulsante RESET.
3. Premere un qualsiasi tasto della tastiera. Sul video appare la scritta:

   YOUR HIGHEST RAM LOCATION IS: XFØØ

   ?

   dove il valore di X dipende dalla quantità di memoria RAM disponibile nel particolare sistema in uso e il "?" sta ad indicare che l'elaboratore è in attesa di comandi.
4. Battere sulla tastiera BC e porre il registratore in "PLAY".

5. Al termine della lettura l'elaboratore stampa un nuovo "?" a capo della successiva riga video. A questo punto battendo il comando GØ vedremo apparire sul video, se la lettura é stata corretta, la scritta "MEMORY SIZE?".
6. In caso di errore di lettura riprendere dal punto 2.

b. Inizializzazione del BASIC.

Qualunque sia il tipo di unità centrale di cui l'utente dispone, abbiamo visto che, al termine del caricamento di un nastro BASIC, appare sul video la scritta "MEMORY SIZE?". Tale domanda è intesa a conoscere lo ammontare di memoria libera che vogliamo assegnare al BASIC per i nostri programmi. La risposta dipenderà ovviamente dalla quantità di memoria RAM di cui dispone il sistema (16K, 2ØK, ecc.). A questo proposito riportiamo di seguito una tabella, che indica le massime dimensioni di memoria assegnabili al BASIC, in risposta alla domanda "MEMORY SIZE?", rapportate appunto all'estensione di memoria RAM del sistema:

Configurazione	Valore
16K	16128
20K	20224
24K	24320
28K	28416
32	32512

e così via secondo la relazione:

Valore=n° bytes di RAM - 256.

In ogni modo l'utente risponde battendo sulla tastiera il numero opportuno ed il tasto di ritorno carrello. Il BASIC chiede allora se desideriamo mantenere in memoria le funzioni seno, coseno, tangente, arcotangente:
WANT SIN-COS-TAN-ATN?
Possiamo rispondere battendo Y per sì oppure N per no ed al solito il ritorno carrello. A questo punto il BASIC indica l'ammontare di memoria libera, in bytes, per il programma dell'utente e l'asterisco per indicare di essere pronto a ricevere ordini.

APPENDICE C

OCCUPAZIONE DI MEMORIA E TEMPI DI ESECUZIONE

a. Occupazione di memoria.

L'ammontare di memoria richiesto per un programma dipende naturalmente dal numero e dal tipo di elementi presenti nel programma stesso. La seguente tabella contieneinformazioni sullo spazio richiesto per i vari elementi di un programma.

Elemento	Spazio richiesto
Variabile intera	5 bytes
" prec. sing.	7 "
" prec. dop.	11 "
" stringa	6 "
Matrice intera	(n° elem.)*2+6+(n° dim.)*2 bytes
" prec. singola	.....4+5.....
" prec. doppia	.....8.....
" stringa	.....3.....
Funzione intrinseca	2 bytes per la chiamata
" utente	6 bytes per la definizione

Parole riservate	1 byte ciascuna, eccetto 2 la ELSE
Altri caratteri	1 byte a testa
Spazio stringhe	1 byte per carattere
Parentesi	6 bytes a coppia
Risultati intermedi	12 bytes
Ciclo FOR attivo	17 bytes
GOSUB	5 bytes

b. Come ottimizzare l'occupazione di memoria.

Lo spazio richiesto per girare un programma può essere notevolmente ridotto, senza influenzare l'esecuzione del programma stesso, seguendo i consigli sotto elencati.

1. Usare più frasi su una stessa linea. Ogni linea richiede difatti 5 bytes per il numero di linea, cosicchè minore è il numero delle linee, minore è l'occupazione di memoria.

2. Eliminare gli spazi non necessari. Ad esempio invece di

```
10 PRINT X, Y, Z
```

scrivere

```
10 PRINTX,Y,Z
```

3. Eliminare le frasi REM, per guadagnare 1 byte per la parola REM e 1 byte per ogni carattere del commento.

4. Usare variabili invece che costanti, specialmente quando il medesimo valore è usato diverse volte. Ad esempio, usare la costante 3.14159 per dieci volte in un programma richiede 40 bytes in più rispetto ad assegnare

```
10 P=3.14159
```

una volta per tutte ed usare P dieci volte.

5. Riusare variabili che non sono più necessarie invece che definirne delle nuove.

6. Far uso di sottoprogrammi piuttosto che riscrivere più volte lo stesso gruppo di istruzioni.

7. Usare gli elementi zero delle matrici. Si ricordi che la matrice dimensionata dall'istruzione DIM A(10) è costituita da undici elementi, da A(0) ad A(10).

8. Usare variabili intere dovunque é possibile.

c. Come ottimizzare i tempi di esecuzione.

1. Eliminando gli spazi e le frasi REM si ottiene una piccola ma sensibile diminuzione nel tempo di esecuzione.

2. Le variabili sono inserite in una tavola nell'ordine della loro prima apparizione all'interno del programma. Quando poi in un programma si fa riferimento ad una variabile, il BASIC ricerca quella variabile nella tabella. Le variabili che si trovano in testa alla tabella vengono ricercate in minor tempo che non quelle in coda. Perciò è preferibile riusare i nomi di variabile e mantenere la lista della variabili più corta possibile.

3. Usare NEXT senza la variabile indice.

4. Utilizzare, dovunque possibile, variabili intere.

APPENDICE D

FUNZIONI MATEMATICHE

Riportiamo qua un elenco di funzioni matematiche, che, sebbene non intrinseche al Child BASIC, possono essere facilmente calcolate facendo uso di funzioni BASIC esistenti.

La funzione:	si calcola come:
SECANTE	SEC(X) = 1/COS(X)
COSECANTE	CSC(X) = 1/SIN(X)
COTANGENTE	COT(X) = 1/TAN(X)
ARCOSENO	ARCSIN(X) = ATN(X/SQR(-X*X+1))
ARCOCOSENO	ARCCOS(X) = -ATN(X/SQR(-X*X+1)) 1.5708
SENO IPERBOLICO	SINH(X) = (EXP(X)-EXP(-X))/2
COSENO IPERBOLICO	COSH(X) = (EXP(X)+EXP(-X))/2
TANGENTE IPERBOLICA	TANH(X) = 1-2*EXP(-X)/(EXP(X)+EXP(-X))
ARCOSENO IPERBOLICO	ARCSINH(X) = LOG(X+SQR(X*X+1))
ARCOCOSENO IPERBOLICO	ARCCOSH(X) = LOG(X+SQR(X*X-1))
ARCOTANGENTE IPERBOLICA	ARCTANH(X) = LOG((1+X)/(1-X))/2

APPENDICE E

BASIC E LINGUAGGIO MACCHINA

Il Child Extended BASIC permette un facile interfacciamento con sottoprogrammi utente scritti in linguaggio macchina. Questo si ottiene tramite la USR che consente ai programmi BASIC di chiamare sottoprogrammi macchina allo stesso modo delle funzioni BASIC.

Il primo passo da farsi nel predisporre una routine in linguaggio macchina per un programma BASIC é quello di preservare dello spazio di memoria per tale routine. Per questo, quando durante la fase di inizializzazione il BASIC chiede "MEMORY SIZE?", la risposta deve essere l'ammontare di memoria disponibile meno lo spazio richiesto dal sottoprogramma utente (o dai sottoprogrammi utente). Il BASIC usa tutti i bytes che trova dalla locazione zero in su, per cui solo le locazioni di memoria più alte possono essere riservate a questi sottoprogrammi. Se la risposta alla domanda "MEMORY SIZE?" è un valore troppo piccolo, il BASIC ripeterà la sua richiesta, fintanto che non ottiene tutta la memoria di cui necessita.

E-2

Le routine in linguaggio macchina possono essere caricate in memoria da tastiera o dal pannello RTOC (a seconda della versione di unità centrale a disposizione) o anche tramite un programma BASIC per mezzo dell'istruzione POKE.

La funzione USR effettua una chiamata al sottoprogramma il cui indirizzo è quello esplicitamente definito nella DEFUSR corrispondente (v. par. 5-3). Se una USR viene chiamata senza che sia stato caricato nessun sottoprogramma utente, viene segnalato l'errore ILLEGAL FUNCTION CALL.

Quando la USR è chiamata, lo stack pointer è predisposto per otto livelli (16 bytes) di stack. Se sono necessari più livelli, lo stack del BASIC può essere salvato ed un nuovo stack usato dal sottoprogramma macchina. Ovviamente, prima di ritornare dal sottoprogramma, è necessario ripristinare lo stack del BASIC.

Un sottoprogramma utente può alterare tutta la memoria e tutti i registri, anche se, naturalmente, non dovrebbe toccare le locazioni di memoria interne al BASIC, nè dovrebbe estrarre dallo stack più bytes di quan-

ti ne abbia immessi.

La funzione USR viene chiamata con un singolo argomento. Tale argomento é passato alla routine utente tramite la coppia di registri HL, il cui contenuto è un puntatore al valore dell'argomento (v. par. 5-3).

Per passare indietro un risultato da un sottoprogramma, è necessario caricarne il valore (un intero con segno a 16 bits) nella coppia HL e chiamare poi la routine MAKINT, il cui indirizzo è memorizzato nelle locazioni 6 e 7. Se questa routine non viene chiamata, USR(X) ritorna X. Per ritornare sotto il BASIC il sottoprogramma utente esegue allora un'istruzione RET.

Si tenga comunque presente che ulteriori argomenti possono essere passati al sottoprogramma tramite la frase POKE e ritornati tramite la PEEK.

Si ricorda infine che possono essere chiamati fino a dieci sottoprogrammi, mediante le funzioni USR0, USR1,..., USR9.

I N D I C E

Capitolo I	- INTRODUZIONE E GENERALITA'	Pag.	1-1
Par. 1-1	- Introduzione al manuale	"	1-1
a	- Convenzioni	"	1-1
b	- Definizioni	"	1-2
Par. 1-2	- Modi di funzionamento	"	1-4
Par. 1-3	- Formati	"	1-5
a	- Linee	"	1-5
b	- Commenti	"	1-10
c	- Errori	"	1-11
Par. 1-4	- Primi elementi di Editing	"	1-12
a	- Correzione di singoli caratteri	"	1-13
b	- Correzione di linee	"	1-14
c	- Correzione di programmi	"	1-14
Capitolo II	- ESPRESSIONI E FRASI	"	2-1
Par. 2-1	- Espressioni	"	2-1
a	- Costanti	"	2-1
b	- Variabili	"	2-4
c	- Matrici	"	2-8
d	- Operatori e precedenze	"	2-11
e	- Operatori logici	"	2-13
f	- La frase LET	"	2-16
Par. 2-2	- Salti, cicli e sottoprogrammi	"	2-17
a	- Salti (GOTO, IF...THEN, ON...GOTO)	"	2-17
b	- Cicli (FOR, NEXT)	"	2-22

c - Sottoprogrammi (GOSUB, RETURN) Pag. 2-25
d - Errori di OUT OF MEMORY " 2-27

Par. 2-3 - Ingresso/Uscita " 2-27
a - INPUT " 2-27
b - PRINT " 2-29
c -DATA, READ, RESTORE " 2-30
d - CSAVE*, CLOAD* " 2-33
e - Miscellanee (WAIT, PEEK, POKE, " 2-35
OUT, INP)

Capitolo III - FUNZIONI " 3-1
Par. 3-1 - Funzioni intrinseche " 3-1
Par. 3-2 - Funzioni definite dall'utente " 3-2
Par. 3-3 - Errori nelle chiamate " 3-3

Capitolo IV - STRINGHE " 4-1
Par. 4-1 - Dati alfanumerici " 4-1
Par. 4-2 - Operazioni sulle stringhe " 4-3
Par. 4-3 - Funzioni alfanumeriche " 4-5

Capitolo V - ALCUNI IMPORTANTI ASPETTI " 5-1
DEL CHILD EXTENDED BASIC
Par. 5-1 - Frasi " 5-1
a - ERASE " 5-1
b - LINE INPUT " 5-2

c - SWAP Pag. 5-3
d - TRON, TROFF " 5-4
e - IF...THEN...ELSE " 5-5
f - DEFINT, DEFSNG, DEFDBL, DEFSTR " 5-5
g - WIDTH " 5-6
h - Trappole per errori " 5-6

Par. 5-2 - Operatori " 5-13
a - Divisione intera " 5-13
b - Modulo aritmetico (MOD) " 5-14

Par. 5-3 - Funzioni " 5-14
a - Funzioni intrinseche " 5-14
b - La frase DEFUSR " 5-14

Par. 5-4 - Il comando EDIT " 5-19

Par. 5-5 - La frase PRINT USING " 5-32

Capitolo VI - LISTE ED ELENCHI " 6-1

Par. 6-1 - Comandi " 6-1

Par. 6-2 - Frasi " 6-7

Par. 6-3 - Funzioni intrinseche " 6-17

Par. 6-4 - Caratteri speciali " 6-27

Par. 6-5 - Messaggi di errore " 6-32

Par. 6-6 - Parole riservate " 6-39

IV

APPENDICE A - Il set dei caratteri ASCII

APPENDICE B - Come caricare ed inizializzare il BASIC

APPENDICE C - Occupazione di memoria e tempi di esecuzione

APPENDICE D - Funzioni matematiche

APPENDICE E - BASIC e linguaggio macchina

General Processor
Via Panciatichi 40 - FIRENZE